

**UNCLASSIFIED**

# **XML Data Flow Configuration File Format Specification**

**Version 1.2.11.1  
19 February 2009**

**For Questions/Comments please contact:**

**[bray\\_dfcf@nsa.gov](mailto:bray_dfcf@nsa.gov)**

**or**

Boyd Fletcher  
SPAWAR SC Pacific  
115 Lake View Parkway  
Suffolk, VA 23435  
757.203.3290  
[boyd.fletcher@navy.mil](mailto:boyd.fletcher@navy.mil)

DISTRIBUTION STATEMENT A

Approved for public release; distribution is unlimited.

**UNCLASSIFIED**

# UNCLASSIFIED

## Table of Contents

1. INTRODUCTION .....	6
2. DOCUMENT AUDIENCE AND USAGE.....	7
3. ACKNOWLEDGEMENTS .....	7
4. DATA FLOW CONFIGURATION (DFC) FILE.....	7
5. DFC.XML .....	9
5.1 DATA FLOW METADATA .....	13
5.1.1 <name> element.....	13
5.1.2 <uuid> element.....	13
5.1.3 <version> element.....	13
5.1.4 <created> element.....	14
5.1.5 <createdBy> element.....	14
5.1.6 <modified> element.....	14
5.1.7 <modifiedBy> element .....	14
5.1.8 <description> element.....	14
5.1.9 <ddms:security> element.....	15
5.1.10 <expirationDate> element.....	15
5.1.11 <enableDate> element.....	15
5.1.12 <ticketRequest> element .....	16
5.1.13 Data Flow Metadata Example .....	16
5.2 GUARDS.....	16
5.2.1 <guard> element.....	16
5.2.2 <options> element (subelement of: guard).....	17
5.2.3 <option> element (subelement of: options) .....	17
5.2.4 <xformsConfig> element (subelement of: guard).....	18
5.2.5 Guards Example .....	18
5.3 DOMAINS .....	19
5.3.1 <domains> element.....	19
5.3.2 <domain> element (subelement of: domains) .....	20
5.3.3 <allowedInboundClassifications> element (subelement of: domain).....	21
5.3.4 <minimumAllowedInboundClassifications> element (subelement of: domain).....	22
5.3.5 <maximumAllowedInboundClassifications> element (subelement of: domain).....	22
5.3.6 <endpoints> element (subelement of: domain) .....	23
5.3.7 <endpoint> element (subelement of: endpoints).....	23
5.3.8 Domains Example .....	25
5.4 FILES .....	27
5.4.1 <files> element .....	27
5.4.2 <schemas> element (subelement of: files).....	28
5.4.3 <schema> element (subelement of: schemas) .....	28
5.4.4 <schematrons> element (subelement of: files) .....	29
5.4.5 <schematron> element (subelement of: schematrons) .....	29
5.4.6 <stylesheets> element (subelement of: files) .....	30
5.4.7 <stylesheet> element (subelement of: stylesheets).....	31
5.4.8 <others> element (subelement of: files) .....	31
5.4.9 <other> element (subelement of: others).....	32
5.4.10 Files Example .....	32
5.5 DOMAIN PAIRS .....	34

## UNCLASSIFIED

5.5.1	<domainPairs> element .....	34
5.5.2	<domainPair> element (subelement of: domainPairs).....	38
5.5.3	<filterset> element (subelement of: domainPair) .....	42
5.5.4	<guard> element (subelement of: filterset) .....	46
5.5.5	<options> element (subelement of: guard) .....	46
5.5.6	<option> element (subelement of: options) .....	47
5.5.7	<xformConfig> element (subelement of: guard).....	47
5.5.8	<fromEndpoints> element (subelement of: filterset) .....	48
5.5.9	<endpoint> element (subelement of: fromEndpoints) .....	48
5.5.10	<toEndpoint> element (subelement of: filterset) .....	49
5.5.11	<endpoint> element (subelement of: toEndpoints) .....	49
5.5.12	<requires> element (subelement of: filterset).....	49
5.5.13	<require> element (subelement of: requires) .....	50
5.5.14	<dependencies> element (subelement of: filterset).....	50
5.5.15	<dependency> element (subelement of: dependencies) .....	51
5.5.16	<associates> element (subelement of: filterset).....	51
5.5.17	<associate> element (subelement of: associates).....	52
5.5.18	<filters> element (subelement of: filterset) .....	52
5.5.19	<xmlvalidator> element (subelement of filters) .....	54
5.5.20	<schematron> element (subelement of filters) .....	55
5.5.21	<xsltransformer> element (subelement of filters).....	56
5.5.22	<xslvalidator> element (subelement of filters).....	57
5.5.23	<wordchecker> element (subelement of filters) .....	58
5.5.24	<xmlnormalizer> element (subelement of filters) .....	59
5.5.25	<classificationchecker> element (subelement of filters) .....	60
5.5.26	<viruschecker> element (subelement of filters) .....	61
5.5.27	<digitalsignaturevalidator> element (subelement of filters) .....	62
5.5.28	<digitalsignatureremover> element (subelement of filters) .....	64
5.5.29	<custom> element (subelement of filters) .....	65
5.5.30	<options> element.....	66
5.5.31	<option> element (subelement of: options) .....	67
5.5.32	<dependencies> element.....	67
5.5.33	<dependency> element (subelement of: dependencies) .....	68
5.5.34	<filterDataInput> element.....	68
5.5.35	<i>Domain Pairs Example</i> .....	69
5.6	DIRTY/CLEAN WORDS XML .....	74
5.6.1	<description> element .....	74
5.6.2	<dirtyCleanWordList> element .....	75
5.6.3	<dirtyWords> element (subelement of: dirtyCleanWordList).....	75
5.6.4	<cleanWords> element (subelement of: dirtyCleanWordList).....	76
5.6.5	<ignoreCharacters> element (subelement of: dirtyCleanWordList).....	76
6.	<b>DIGITAL SIGNING</b> .....	78
6.1	<fileEntry> element.....	79
6.2	<signed> element .....	80
6.3	<signedBy> element .....	80
6.4	<signerRole> element .....	80
6.5	<ds:Signature> element .....	81
7.	<b>COMMENTS XML</b> .....	83
7.1	<comment> element (subelement of: comments) .....	84
8.	<b>CHANGE LOG</b> .....	86

## UNCLASSIFIED

8.1 <i>&lt;message&gt; element</i> .....	87
<b>9. XFORMS (GUARD CUSTOMIZATION INFORMATION) .....</b>	<b>88</b>
9.1    DFC MODEL.....	88
9.2    GUARD-SPECIFIC PROPERTIES .....	88
<b>APPENDIX A - SCHEMAS .....</b>	<b>89</b>
DFC.XSD.....	89
DIGITALSIGNINGS.XSD .....	104
DIRTYCLEANWORDS.XSD .....	106
CHANGELOG.XSD .....	107
<b>APPENDIX B - REFERENCES .....</b>	<b>108</b>

**UNCLASSIFIED****Revision History**

Date	Version	Description	Author
9/17/2007	1.0	Initial feedback version	Joel Schilling
12/10/2007	1.1	Finalized feedback, and added real DFC examples	Joel Schilling
1/23/2008	1.1.1	Added introduction and other clarifications	Boyd Fletcher
2/15/2008	1.2.1	Changes based on feedback from S. Duncan and C. Snapp. Changes major restructuring of filtering logic, removal of root schema concepts, dw/cw changes.	Joel Schilling
2/17/2008	1.2.2	Formatting changes and improved descriptions	Boyd Fletcher
2/19/2008	1.2.3	Minor edits	Boyd Fletcher
3/10/2008	1.2.4	Changed <flow> to <domainPair>, added <fromEndpoints> and <toEndpoint> to filtersets, added type and certificate to <endpoint>, added <associates> to filtersets, changed <signStatus> to <signerRole>.	Boyd Fletcher
3/12/2008	1.2.5	Minor edits/clarifications	B. Fletcher, A. Aquino
3/14/2008	1.2.6	Add ed<option> to the main <guard> properties element, edits to various filterset sub elements.	B. Fletcher
3/18/2008	1.2.7	Minor edits/clarifications	C. Van Stedum
4/1/2008	1.2.8	Clarification Normalization and Digital Signature Validator/Removal sections. Updated References. Updated filter definition.	B. Fletcher
4/3/2008	1.2.9	Add <others> to <files>. Add <filterDataInput> to each of the filter types. Add <urls> to <digtalsignaturevalidator>. Make only one <guard> allowable per <filterset>	B. Fletcher, J. Schilling
8/27/2008	1.2.10	Added <ticketRequest>. Added <xslvalidator>. Changed <endpoints> maxOccurs to 1. Changed <releasableClassifications> to <allowedInboundClassifications>, <minimumClassifications> to <minimumAllowedInboundClassifications>, and <maximumClassifications> to <maximumAllowedInboundClassifications>. Made all attributes on <endpoint> optional except "id". Changed <schema> type enumeration value "w3c" to "W3C". Added <description> to each of the filters.	J. Schilling
10/9/2008	1.2.10	Fixed minor problems found while working on validation logic	J. Schilling
12/18/2008	1.2.11	Added "file" to list of protocols for <endpoint>	J. Schilling
2/19/2009	1.2.11.1	Fixed various schema errors	J. Schilling

## UNCLASSIFIED

### 1. Introduction

The eXtensible Markup Language (XML) Data Flow Configuration File (DFCF) format specification was developed to provide a common format for defining, validating, and approving XML Data Flows for use on XML Cross Domain Guarding Solutions. The DFCF specification has an accompanying application called Bray that can be used to create, edit, and validate DFCFs. (However, the use of the Bray application is not required to create, edit and validate DFCFs.)

The DFCF specification and Bray application were designed to address the following high-level requirements:

- Provide a guard-independent method of describing XML Data Flows
- Reduce guard installation, guard configuration, and site security testing and evaluation (ST&E) time and costs
- Decrease the approval time of a user's XML Data Flows by Intelligence Community (IC)/Department of Defense (DOD) certification and accreditation organizations (i.e. DSAWG/CBTAB, etc.)
- Reduce the administrative burden on IC/DOD Certification and Accreditation (C&A) organizations by providing a consistent way of describing and approving XML Data Flows
- Provide a consistent way to describe XML Data Flows in System Security Authorization Agreements (SSAAs)
- Provide a mechanism to capture comments/questions related to the Data Flow
- Provide a method to increase the assurance that only approved schemas are being loaded onto guards and that the schemas have not been modified in transit
- Provide a graphical tool for locking down and evaluating the security risk of XML Schema Definition files and XML Stylesheet files
- Provide a graphical tool for creating, editing, and validating XML Data Flows
- Provide a graphical tool that can be used by Guard Security Administrators, IC/DOD C&A Staff, and Application Developers
- Provide an API to access the XML Data Flow configuration that can be used in guard development

An XML Data Flow Configuration File uses the zip file format to store the multitude of files that make up a DFCF. This list of files includes the XML documents described by this specification, the Data Flow's schema definition and stylesheet files, dirty/clean word lists, guard XForms, and many others.

The DFCF specification focuses on the following areas of an XML Data Flow:

- Metadata about the flow
- List of Classification Labels
- List of Domain Pairings (i.e. JWICS,SIPRnet; SIPRnet,NIPRnet)
- XML Schema file(s) (.xsd, .rng)
- XML Stylesheet Transformation file(s) (.xslt)
- Schematron Assertion file(s)
- Filter Configuration, Custom Filter Definition, and Filter Ordering
- Clean/Dirty Word Lists in UTF8 format
- Guard Specific Configuration Data via XForms (if required)
- A Digitally Signed Data Flow

## UNCLASSIFIED

### **2. Document Audience and Usage**

This document is primarily intended to be used by guard developers for implementing DFCF into their devices. The Bray application is intended as the primary interface to application developers, evaluators, and certifiers/approver when creating, evaluating, and approving DFCFs. However, it is strongly recommended that anyone involved in creating, evaluating, and approving DFCFs read the descriptions in the DFCF Specification as there are clarifications on usage/interpretation and best practices are described.

### **3. Acknowledgements**

This work would not have been possible without the support of the National Security Agency's Information Assurance Directorate. In particular the authors would like to thank Mark Roberts, Alonza Mumford, and Terence Sanders. The quality and completeness of this specification would not have been possible without the extensive contributions from Stephen Duncan (Northrop Grumman), Chris Snapp (Northrop Grumman), Larry Brown (Lockheed Martin), Russ Hawkins (Lockheed Martin), John Woodruff (AFRL), Dave Amos (Mitre), and Joel Schilling (Dataline).

### **4. Data Flow Configuration (DFC) File**

A DFC file is a ZIP file that contains all the files required to describe an XML Data Flow. The DFC Zip file uses the file extension **.DFC**. The ZIP file specification can be found at <http://www.pkware.com/documents/casestudies/APPNOTE.TXT>. The ZIP file format was chosen because it provides a well understood, compressed, and commonly implemented mechanism for packaging files and directories. Many file formats like Java JAR files and OpenDocument and OpenOffice XML office formats are actually ZIP formatted files. The following table contains the required and optional files and directories that are part of a DFC ZIP file.

Filename & Path	Required	Description
dfc.xml	Yes	The main XML Data Flow configuration file. This specification describes this file in detail.
comments.xml	No	Contains a time-ordered list of comments, which may be individually digitally signed. Comments provide a simple means to communicate and track workflow-type information between a developer, submitter, evaluator, and approver of an XML Data Flow. Some examples of possible uses are: it can be used to explain why a schema rating is high and how it is being mitigated by a filter or another function in a guard; it can be used by an evaluator in seeking clarification from the developer on why the schema has a

**UNCLASSIFIED**

Filename & Path	Required	Description
		particular set of elements; and it can be used by an approver to state stipulations on how the Data Flow can be used.
digitalsignings.xml	No	Contains a digitally signed checksum of all the files in the DFCF. A digitally signed DFCF provides a mechanism to guarantee the integrity of the DFCF while in transit and is used by the approver to certify the DFCF is allowed to be installed.
changelog.xml	Yes	Contains a time ordered list of all changes to the DFCF.
bray.xml	Yes	Used by the Bray application for application-specific configuration and state information. It is not used by a guard when loading a DFCF.
schemas/	Yes	Directory containing all schema files and all included subfiles.
schematrons/	Yes	Directory containing all schematron files.
stylesheets/	Yes	Directory containing all stylesheet files.
others/	No	Directory containing all files other than schemas, schematrons, and stylesheets.
certificates/	No	Directory containing PKI PEM certificates used for Endpoint authentication.
guardconfig/	Yes	Directory containing all XForms form files and associated stylesheets. Also contains the instance files for DFC Guard Properties.
domainPairs/	Yes	Contains directories for each Domain Pair.
domainPairs/domain1_domain2 /	Yes	Domain Pairs directories. The name of each directory is derived from domain1 and domain2, separated by an underscore. Examples: /domainPairs/nipr_sipr,
domainPairs/domain1_domain2 /filterset/	Yes	Filterset directories. Each Domain Pair directory contains directories based on the name of the Filtersets. Examples: /domainPairs/nipr_sipr/dsg-filters, /domainPairs/nipr_sipr/dfg-filters2
domainPairs/domain1_domain2 /filterset/guardconfig/	Yes	Guard Configuration directory. Each Filterset directory contains a directory named "/guardconfig" containing the instance files for

## UNCLASSIFIED

Filename & Path	Required	Description
		Filterset Guard Properties. Examples: /domainPairs/nipr_sipr/dsg-filters1/guardconfig, /domainPairs/nipr_sipr/dsg-filters2/guardconfig
Wordcheck filter directories	No	Each Filterset directory contains directories for the wordcheck filters, based on the filter ID. It contains “dirty-clean-words.xml” and all text files containing dirty/clean words lists. Examples: /domainPairs/nipr_sipr/dsg-filters1/1, /domainPairs/nipr_sipr/dsg-filters2/3
Custom filter directories	No	Each Filterset directory contains directories for the custom filters, based on the filter ID. It contains all files associated with the filter. Examples: /domainPairs/nipr_sipr/dsg-filters2/7, /domainPairs/nipr_sipr/dsg-filters2/8

## 5. dfc.xml

The dfc.xml file is the main configuration file for an XML Data Flow. It contains the Data Flow metadata; guard info; filter configuration; references to the Data Flow's XML schemas, stylesheets, and schematron files; classification labels; domains; domain pairs; and other miscellaneous data. It is located at the root level in the DFC ZIP. The root element is `<dfc>`. The required `version` attribute defines the version of the DFCF specification that is intended to be used to interpret the DFC ZIP. For this version of the specification, the value of version will be “1.2.11”

schema:

```
<?xml version="1.0"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:ddms="http://metadata.dod.mil/mdr/ns/DDMS/1.3/"
  xmlns:dfc="http://cie.jfcom.mil/schema/dfc"
  targetNamespace="http://cie.jfcom.mil/schema/dfc"
  elementFormDefault="qualified">
  <xs:import
    namespace="http://metadata.dod.mil/mdr/ns/DDMS/1.3/"
    schemaLocation="http://metadata.dod.mil/mdr/ns/DDMS/1.3/" />

  <xs:element name="dfc">
    <xs:complexType>
      <xs:sequence>
        [Elements omitted]
      </xs:sequence>
    <xs:attribute name="version" use="required">
```

**UNCLASSIFIED**

```
<xs:simpleType>
  <xs:restriction base="xs:string">
    <xs:enumeration value="1.2.10" />
    <xs:enumeration value="1.2.11" />
  </xs:restriction>
</xs:simpleType>
</xs:attribute>
</xs:complexType>

<xs:key name="domainKey">
  <xs:selector xpath="dfc:domains/dfc:domain"/>
  <xs:field xpath="@id"/>
</xs:key>

<xs:keyref name="domain1Ref" refer="dfc:domainKey">
  <xs:selector xpath="dfc:domainPairs/dfc:domainPair"/>
  <xs:field xpath="@domain1Ref"/>
</xs:keyref>

<xs:keyref name="domain2Ref" refer="dfc:domainKey">
  <xs:selector xpath="dfc:domainPairs/dfc:domainPair"/>
  <xs:field xpath="@domain2Ref"/>
</xs:keyref>

<xs:keyref name="fromDomainRef" refer="dfc:domainKey">
  <xs:selector
    xpath="dfc:domainPairs/dfc:domainPair/dfc:filterset"/>
  <xs:field xpath="@fromDomainRef"/>
</xs:keyref>

<xs:keyref name="toDomainRef" refer="dfc:domainKey">
  <xs:selector
    xpath="dfc:domainPairs/dfc:domainPair/dfc:filterset"/>
  <xs:field xpath="@toDomainRef"/>
</xs:keyref>

<xs:key name="endpointKey">
  <xs:selector xpath="dfc:domains/dfc:domain/dfc:endpoints/
    dfc:endpoint" />
  <xs:field xpath="@id" />
</xs:key>
</xs:element>
<xs:key name="schemaKey">
  <xs:selector xpath="dfc:files/dfc:schemas/dfc:schema" />
  <xs:field xpath="@id" />
</xs:key>
<xs:key name="schematronKey">
  <xs:selector xpath="dfc:files/dfc:schematrons/
    dfc:schematron" />
  <xs:field xpath="@id" />
</xs:key>
<xs:key name="stylesheetKey">
  <xs:selector xpath="dfc:files/dfc:stylesheets/
    dfc:stylesheet" />
  <xs:field xpath="@id" />
</xs:key>
<xs:key name="otherKey">
```

## UNCLASSIFIED

```
<xs:selector xpath="dfc:files/dfc:others/dfc:other" />
<xs:field xpath="@id" />
</xs:key>
<xs:keyref name="xmlvalidatorKeyref" refer="dfc:schemaKey">
    <xs:selector xpath="dfc:domainPairs/dfc:domainPair/
        dfc:filterset/dfc:filters/dfc:xmlvalidator" />
    <xs:field xpath="@schemaRef" />
</xs:keyref>
<xs:keyref name="schematronKeyref" refer="dfc:schematronKey">
    <xs:selector xpath="dfc:domainPairs/dfc:domainPair/
        dfc:filterset/dfc:filters/dfc:schematron" />
    <xs:field xpath="@schematronRef" />
</xs:keyref>
<xs:keyref name="xsltransformerKeyref"
    refer="dfc:stylesheetKey">
    <xs:selector xpath="dfc:domainPairs/dfc:domainPair/
        dfc:filterset/dfc:filters/dfc:xsltransformer" />
    <xs:field xpath="@stylesheetRef" />
</xs:keyref>
<xs:keyref name="xslvalidatorKeyref"
    refer="dfc:stylesheetKey">
    <xs:selector xpath="dfc:domainPairs/dfc:domainPair/
        dfc:filterset/dfc:filters/dfc:xslvalidator" />
    <xs:field xpath="@stylesheetRef" />
</xs:keyref>

<xs:complexType name="options">
    <xs:sequence>
        <xs:element name="option" minOccurs="1"
            maxOccurs="unbounded">
            <xs:complexType>
                <xs:simpleContent>
                    <xs:extension base="xs:string">
                        <xs:attribute name="name" use="required" />
                    </xs:extension>
                </xs:simpleContent>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="dependencies">
    <xs:sequence>
        <xs:element name="dependency" minOccurs="0"
            maxOccurs="unbounded">
            <xs:complexType>
                <xs:attribute name="ref" type="xs:string"
                    use="required"/>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="filterDataInput">
    <xs:attribute name="location" use="required">
        <xs:simpleType>
            <xs:restriction base="xs:string">
```

## UNCLASSIFIED

```
<xs:enumeration value="dependency" />
<xs:enumeration value="original" />
<xs:enumeration value="none" />
</xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="dependencyFilterRef" type="xs:string"
    use="optional" />
</xs:complexType>
</xs:schema>
```

example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<dgc
    xmlns:IC="urn:us:gov:ic:ism:v2"
    xmlns:ddms="http://metadata.dod.mil/mdr/ns/DDMS/1.3/"
    xmlns="http://cie.jfcom.mil/schema/dgc"
    version="1.2.11">
    [Elements omitted]
</dgc>
```

## 5.1 Data Flow Metadata

### 5.1.1 <name> element

A short human-readable name for the Data Flow. It is commonly used by a guard's administration tools when displaying which Data Flows are configured on a device. It should not be used by a guard to determine the uniqueness of Data Flows. It is recommended that the name only contain alphanumeric characters, dashes, periods, and underscores.

schema:

```
<xs:element name="name" type="xs:string" minOccurs="1"
maxOccurs="1" />
```

example:

```
<name>Biometrics</name>
```

### 5.1.2 <uuid> element

A 128-bit value universally unique identifier (UUID) for the DFC. New versions of the same XML Data Flow will have the same UUID but the version number will be increased. The UUID, when combined with the version, is typically used by systems to provide a guaranteed way to determine uniqueness of an XML Data Flow. Bray creates UUIDs using the Java library class UUID as specified at:

<http://java.sun.com/j2se/1.5.0/docs/api/java/util/UUID.html>

schema:

```
<xs:element name="uuid" type="xs:string" minOccurs="1"
maxOccurs="1" />
```

example:

```
<uuid>75aa38db-9a0b-4cb4-8d0a-f1e95fc761b7</uuid>
```

### 5.1.3 <version> element

The version of the DFCF. There are no restrictions on the value used for the version; however, it is recommend that the MajorRelease.MinorRelease.BugFix (i.e. 4.2.3) format be used. The version, when combined with the UUID, is typically used by systems to provide a guaranteed way to determine uniqueness of an XML Data Flow.

schema:

```
<xs:element name="version" type="xs:string" minOccurs="1"
maxOccurs="1" />
```

example:

```
<version>1.0</version>
```

## UNCLASSIFIED

### 5.1.4 <created> element

The date the DFCF was created (format: CCYY-MM-DD)

schema:

```
<xs:element name="created" type="xs:date" minOccurs="1"  
maxOccurs="1"/>
```

example:

```
<created>2008-01-01</created>
```

### 5.1.5 <createdBy> element

The author of the DFC ZIP. Bray uses the operating system-specific username format to populate this element.

schema:

```
<xs:element name="createdBy" type="xs:string" minOccurs="1"  
maxOccurs="1"/>
```

example:

```
<createdBy>snappc</createdBy>
```

### 5.1.6 <modified> element

The date the DFC ZIP was last modified (format: CCYY-MM-DD)

schema:

```
<xs:element name="modified" type="xs:date" minOccurs="1"  
maxOccurs="1"/>
```

example:

```
<modified>2008-01-04</modified>
```

### 5.1.7 <modifiedBy> element

The name of the last user to modify the DFC ZIP. Bray uses the operating system-specific username format to populate this element.

schema:

```
<xs:element name="modifiedBy" type="xs:string" minOccurs="1"  
maxOccurs="1"/>
```

example:

```
<modifiedBy>duncans</modifiedBy>
```

### 5.1.8 <description> element

This optional element contains the text description of the DFC. It is strongly recommended that users populate this description field since the evaluators and

## UNCLASSIFIED

approvers use this information in determining its suitability for the environment in which it is being deployed. The description typically answers the “who, what, where, and why” questions.

schema:

```
<xs:element name="description" type="xs:string" minOccurs="0" maxOccurs="1"/>
```

example:

```
<description>This is the DFC for Biometrics demo</description>
```

### 5.1.9 <ddms:security> element

The classification level for the entire DFC zip and all files included in it. This element complies with the Department of Defense Metadata Standard (DDMS) schema and uses the Intelligence Community Information Security Marking Implementation Guide (IC ISM IG).

schema:

```
<xs:element ref="ddms:security" minOccurs="1" maxOccurs="1"/>
```

example:

```
<ddms:security IC:ownerProducer="USA" IC:classification="U" />
```

### 5.1.10 <expirationDate> element

This optional element provides an expiration date for guards to know when the Data Flow is no longer valid, and to disable it at that time.

schema:

```
<xs:element name="expirationDate" type="xs:date" minOccurs="0" maxOccurs="1"/>
```

example:

```
<expirationDate>2010-01-01</expirationDate>
```

### 5.1.11 <enableDate> element

This optional element provides an enable date for guards to know when the Data Flow is valid to activate.

schema:

```
<xs:element name="enableDate" type="xs:date" minOccurs="0" maxOccurs="1"/>
```

example:

```
<enableDate>20008-01-01</enableDate>
```

## UNCLASSIFIED

### 5.1.12 <ticketRequest> element

This optional element allows the user to enter the ticket or request number for the DFC for tracking purposes.

schema:

```
<xs:element name="ticketRequest" type="xs:string" minOccurs="0" maxOccurs="1"/>
```

example:

```
<ticketRequest>T-800</ticketRequest>
```

### 5.1.13 Data Flow Metadata Example

Here is an example XML file including Data Flow Metadata:

example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<dfc
  xmlns:IC="urn:us:gov:ic:ism:v2"
  xmlns:ddms="http://metadata.dod.mil/mdr/ns/DDMS/1.3/"
  xmlns="http://cie.jfcom.mil/schema/dfc"
  version="1.2.11">
  <name>Biometrics</name>
  <uuid>75aa38db-9a0b-4cb4-8d0a-f1e95fc761e7</uuid>
  <version>1.0</version>
  <created>2008-01-01</created>
  <createdBy>snappc</createdBy>
  <modified>2008-01-04</modified>
  <modifiedBy>duncans</modifiedBy>
  <description> This is the DFC for Biometrics demo</description>
  <ddms:security IC:ownerProducer="USA" IC:classification="U" />
  <expirationDate>2010-01-01</expirationDate>
  <enableDate>2008-01-01</enableDate>
  <ticketRequest>2008-01-01</ticketRequest>
</dfc>
```

## 5.2 Guards

### 5.2.1 <guard> element

This is an optional element that defines the **Guards** that will be associated with the Data Flow. The required `name` and `version` attributes define the guard's symbolic name and version. Each guard can have optional Name/Value elements or an optional **XForms Configuration** for configuring guard-specific properties (like networking settings).

The optional `nativeSupport` attribute is a boolean value defining whether the guard supports DFC natively.

There is a `generic` guard value that is used when a DFCF author does not want to limit a DFCF to a specific guard's features/capabilities. This essentially makes the DFCF guard-independent, but care must be taken to make sure that the filter rules are not

## UNCLASSIFIED

incompatible with the basic capabilities of most XML guards.

schema:

```
<xs:element name="guard" minOccurs="0" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      [See 5.2.2 for definition of options element]
      [See 5.2.4 for definition of xformsConfig element]
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="version" type="xs:string"
      use="required"/>
    <xs:attribute name="nativeSupport" type="xs:boolean"
      use="optional"/>
  </xs:complexType>
  <xs:key name="guardOptionsKey">
    <xs:selector xpath="dfc:options/dfc:option" />
    <xs:field xpath="@name" />
  </xs:key>
</xs:element>
```

example:

```
<guard name="WSG" version="1.0" />
<guard name="DSG" version="3.0">
  <xformConfig
    xformss="guardconfig/dsg-3.0.xhtml"
    css="guardconfig/dsg-3.0.css"
    instance="guardconfig/dsg-3.0-instance.xml" />
</guard>
```

### 5.2.2 *<options> element (subelement of: guard)*

This optional element provides a way to configure simple guard-specific information without the overhead of an XForms form being defined.

schema:

```
<xs:complexType name="options">
  <xs:sequence>
    [See 5.2.3 definition of option element]
  </xs:sequence>
</xs:complexType>
```

example:

```
<options>
  <option name="max_messages_per_second">100</option>
</options>
```

### 5.2.3 *<option> element (subelement of: options)*

This element contains one or more name/value pairs. The required `name` attribute must be unique within the scope of the **Guard**. The text element of `<option>` contains the value, allowing Unicode characters to be included in the value.

## UNCLASSIFIED

schema:

```
<xs:element name="option" minOccurs="1" maxOccurs="unbounded">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="name" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

example:

```
<option name=" max_messages_per_second ">100</option>
```

### 5.2.4 *<xformsConfig> element (subelement of: guard)*

This element defines the optional **Xforms Configuration** for a **Guard**. The required `xforms` attribute defines the path to the XForms form file. The optional `css` attribute contains the path to the stylesheet to decorate the XForms form. The required `instance` attribute contains the path to the guard properties file that is generated after the user completes the XForms form. The default path for these attributes is “guardconfig/”.

DFCF 1.2.11-based XForms must be compliant with version 1.0 of World Wide Web Consortium’s XForms specification (<http://www.w3.org/MarkUp/Forms>).

This feature provides a mechanism for guard developers to prompt the user for specific information that is unique to that guard and is required in order to properly configure it.

schema:

```
<xs:element name="xformConfig" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:attribute name="xforms" type="xs:string" use="required"/>
    <xs:attribute name="css" type="xs:string" use="optional"/>
    <xs:attribute name="instance" type="xs:string"
      use="required"/>
  </xs:complexType>
</xs:element>
```

example:

```
<xformConfig
  xforms="guardconfig/dsg-3.0.xhtml"
  css="guardconfig/dsg-3.0.css"
  instance="guardconfig/dsg-3.0-instance.xml" />
```

### 5.2.5 *Guards Example*

Cumulative example, including **Data Flow Metadata** and **Guards**:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

## UNCLASSIFIED

```
<dfc
  xmlns:IC="urn:us:gov:ic:ism:v2"
  xmlns:ddms="http://metadata.dod.mil/mdr/ns/DDMS/1.3/"
  xmlns="http://cie.jfcom.mil/schema/dfc"
  version="1.2.11">

  <name>Biometrics</name>
  <uuid>75aa38db-9a0b-4cb4-8d0a-f1e95fc761e7</uuid>
  <version>1.0</version>
  <created>2008-01-01</created>
  <createdBy>snappc</createdBy>
  <modified>2008-01-04</modified>
  <modifiedBy>duncans</modifiedBy>
  <description>This is the DFC for Biometrics demo</description>
  <ddms:security IC:ownerProducer="USA" IC:classification="U" />
  <expirationDate>2010-01-01</expirationDate>
  <enableDate>2008-01-01</enableDate>
  <ticketRequest>2008-01-01</ticketRequest>

  <guard name="WSG" version="1.0" />
  <guard name="DSG" version="3.0">
    <options>
      <option name="max_messages_per_second">100</option>
    </options>
    <xformConfig
      xforms="guardconfig/dsg-3.0.xhtml"
      css="guardconfig/dsg-3.0.css"
      instance="guardconfig/dsg-3.0-instance.xml" />
  </guard>
</dfc>
```

### 5.3 Domains

#### 5.3.1 <domains> element

The list of the **Domains** for the Data Flow. A domain is defined as a set of systems (may include networks) grouped under a single security policy. Some examples of domains would be SIPRnet, JWICS, NIPRnet, and CENTRIXS GCTF.

schema:

```
<xs:element name="domains" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:sequence>
      [See 5.3.2 for definition of domain element]
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

example:

```
<domains>
  <domain id="nipr">
    <description>This is the NIPR domain</description>
    <allowedInboundClassifications>
      <ddms:security IC:ownerProducer="USA"
```

## UNCLASSIFIED

```
    IC:classification="U" />
  </allowedInboundClassifications>
  <endpoints>
    <endpoint
      id="NIPRWebService"
      protocol="https"
      ipAddress="100.101.102.103"
      sendPort="8080"
      type="server"
      pathInfo="/service/WebService"
      certificate="certificates/sipr-cert.pki">
      <description>Web service on the NIPR
        network</description>
    </endpoint>
  </endpoints>
</domain>
<domain id="sipr">
  <description>This is the SIPR domain</description>
  <minimumAllowedInboundClassifications>
    <ddms:security IC:ownerProducer="USA"
      IC:classification="U" />
  </minimumAllowedInboundClassifications>
  <maximumAllowedInboundClassifications>
    <ddms:security IC:ownerProducer="USA"
      IC:classification="S" />
  </maximumAllowedInboundClassifications>
  <endpoints>
    <endpoint
      id="SIPRWebService"
      protocol="https"
      ipAddress="200.201.202.203"
      sendPort="8080"
      type="server"
      pathInfo="/service/WebService" />
  </endpoints>
</domain>
</domains>
```

### 5.3.2 <domain> element (subelement of: domains)

The <domain> element defines the structure of a domain including its allowed classification labels and its endpoints. The required id attribute must be unique, and defines the domain's symbolic token name, and must contain only the following characters: alphanumeric, dashes, and underscores. The domain id must be 64 characters or less in length. (See schema from section 5 for the schema xs:key definition) Because of the way that filterset directories are created, the id must also be unique in a non-case sensitive manner. There is an optional <description> element that can be used to provide a text description of the domain. The <domain> element contains the **Classifications** and **Endpoints** elements.

schema:

```
<xss:element name = "domain" minOccurs="2" maxOccurs="unbounded">
  <xss:complexType>
    <xss:sequence>
      <xss:element name="description" minOccurs="0"
        maxOccurs="1" />
```

## UNCLASSIFIED

```
[See 5.3.3 for allowedInboundClassifications element]
[See 5.3.4 for minimumAllowedInboundClassifications
element]
[See 5.3.5 for maximumAllowedInboundClassifications
element]
[See 5.3.6 for definition of endpoints element]
</xs:sequence>
<xs:attribute name="id" type="xs:string" use="required"/>
</xs:complexType>
</xs:element>
```

example:

```
<domain id="nipr">
  <description>This is the NIPR domain</description>
  <allowedInboundClassifications>
    <ddms:security IC:ownerProducer="USA"
      IC:classification="U" />
  </allowedInboundClassifications>
  <endpoints>
    <endpoint
      id="NIPRWebService"
      protocol="https"
      ipAddress="100.101.102.103"
      sendPort="8080"
      type="server"
      pathInfo="/service/WebService"
      certificate="certificates/sipr-cert.pki">
      <description>Web service on the NIPR network</description>
    </endpoint>
  </endpoints>
</domain>
```

### 5.3.3 <allowedInboundClassifications> element (subelement of: domain)

This element defines the list of allowed inbound **Classifications** for the **Domain**. Each **Classification** is a DDMS Security-compliant classification label defining an allowed inbound classification label for this **Domain** and all **Endpoints** contained within this **Domain**. For a thorough understanding of how security markings in the DOD and IC are constructed it is strongly recommended that guard developers and users read the “Intelligence Community Metadata Standards for Information Assurance - Information Security Marking Implementation Guide: Release 2.0” and its associated schemas that were published on 30 April 2004 by the Intelligence Community Metadata Working Group. The list of classification labels is in no particular order and there is not an implied dominance relationship between them. The **allowedInboundClassifications** element is typically used in environments where the classification label checks are a literal comparison (i.e. it either matches or it does not) and not used in environments where dominance is important. However, there is nothing strictly prohibiting a dominance check if the guard supports that type of validation.

schema:

```
<xs:element name="allowedInboundClassifications" minOccurs="0"
  maxOccurs="1">
</xs:complexType>
```

## UNCLASSIFIED

```
<xs:sequence>
    <xs:element ref="ddms:security" minOccurs="1"
        maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
</xs:element>
```

example:

```
<allowedInboundClassifications>
    <ddms:security IC:ownerProducer="USA" IC:classification="U" />
    <ddms:security IC:ownerProducer="USA" IC:classification="S" />
    <ddms:security IC:ownerProducer="USA" IC:classification="S"
        IC:disseminationControls="REL"
        IC:releasableTo="USA GBR AUS" />
</allowedInboundClassifications>
```

### 5.3.4 <*minimumAllowedInboundClassifications*> element (*subelement of: domain*)

This optional element provides a mechanism to specify the lowest classification labels allowed for a **Domain**. Contains one or more DDMS Security-compliant classification labels. It should contain either one classification label or multiple classification labels of equal dominance.

schema:

```
<xs:element name="minimumAllowedInboundClassifications"
minOccurs="0"
maxOccurs="1">
<xs:complexType>
    <xs:sequence>
        <xs:element ref="ddms:security" minOccurs="1"
            maxOccurs="unbounded"/>
    </xs:sequence>
</xs:complexType>
</xs:element>
```

example:

```
<minimumAllowedInboundClassifications>
    <ddms:security IC:ownerProducer="USA" IC:classification="U" />
</minimumAllowedInboundClassifications>
```

### 5.3.5 <*maximumAllowedInboundClassifications*> element (*subelement of: domain*)

This optional element provides a mechanism to specify the highest classification labels allowed for a **Domain**. Contains one or more DDMS Security-compliant classification labels. It should contain either one classification label or multiple classification labels of equal dominance.

schema:

```
<xs:element name="maximumAllowedInboundClassifications"
minOccurs="0"
maxOccurs="1">
```

## UNCLASSIFIED

```
<xs:complexType>
  <xs:sequence>
    <xs:element ref="ddms:security" minOccurs="1"
      maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
```

example:

```
<maximumAllowedInboundClassifications>
  <ddms:security IC:ownerProducer="USA" IC:classification="S" />
</maximumAllowedInboundClassifications>
```

### 5.3.6 <endpoints> element (subelement of: domain)

This element defines a list of **Endpoints** for the **Domain**.

schema:

```
<xs:element name="endpoints" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:sequence>
      [See 5.3.7 for definition of endpoint element]
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

example:

```
<endpoints>
  <endpoint
    id="NIPRWebService"
    protocol="https"
    ipAddress="100.101.102.103"
    sendPort="8080"
    type="server"
    pathInfo="/service/WebService"
    certificate="certificates/sipr-cert.pki">
    <description>Web service on the NIPR network</description>
  </endpoint>
</endpoints>
```

### 5.3.7 <endpoint> element (subelement of: endpoints)

The **Endpoint** element is used to describe the clients and servers (collectively known as endpoints) that are connecting to the guard. An endpoint description contains a hostname, IP Address, TCP or UDP ports, protocol and Uniform Resource Locator (URL).

The required `id` attribute defines the symbolic name of the endpoint. Valid characters for

## UNCLASSIFIED

`id` are alphanumeric, dashes, and underscores. The `id` attribute's value must be unique across all **Endpoints** in the **Data Flow** (See schema from section 5 for the schema `xs:key` definition.)

The optional `hostName` attribute defines the host name. Valid characters for `hostName` are alphanumeric and dashes. Underscores are prohibited due to incompatibility with the Domain Name System (DNS).

The optional `ipAddress` attribute defines the endpoint's IP address.

The optional `sendPort` attribute defines the combined/send TCP or UDP port for the endpoint. It should be a non-negative integer value.

The optional `receivePort` attribute defines the receive TCP or UDP port. For example, the `sendPort` would be 443 if the guard is communicating with the endpoint over HTTPS. It should be a non-negative integer value.

The optional `type` attribute identifies the endpoint type. Valid values are `client`, `server`, and `both`.

The optional `protocol` attribute defines the endpoint protocol. Valid values for `protocol` are: `dsg2`, `dsg3`, `dsg4`, `rm`, `isse`, `http`, `https`, `ftp`, `sftp`, `tcp-socket`, `udp-socket`, `file`, and `other`.

The optional `pathInfo` attribute defines any additional endpoint URL path info.

The optional `certificate` attribute contains the full DFCF path to a PEM PKI certificate. The certificate files are by default stored in the "certificates/" directory.

There is an optional `<description>` element allowing a text description to be added to the **Endpoint**.

schema:

```
<xs:element name="endpoint" minOccurs="1" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence minOccurs="0" maxOccurs="1">
      <xs:element name="description" type="xs:string" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required"/>
    <xs:attribute name="hostName" type="xs:string"
      use="optional"/>
    <xs:attribute name="ipAddress" type="xs:string"
      use="optional"/>
    <xs:attribute name="sendPort" type="xs:string"
      use="optional"/>
    <xs:attribute name="receivePort" type="xs:string"
      use="optional"/>
    <xs:attribute name="type" use="optional">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="client"/>
          <xs:enumeration value="server"/>
          <xs:enumeration value="both"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
```

## UNCLASSIFIED

```
</xs:simpleType>
</xs:attribute>
<xs:attribute name="protocol" use="optional">
    <xs:simpleType>
        <xs:restriction base="xs:string">
            <xs:enumeration value="dsg2"/>
            <xs:enumeration value="dsg3"/>
            <xs:enumeration value="dsg4"/>
            <xs:enumeration value="rm"/>
            <xs:enumeration value="isse"/>
            <xs:enumeration value="http"/>
            <xs:enumeration value="https"/>
            <xs:enumeration value="ftp"/>
            <xs:enumeration value="sftp"/>
            <xs:enumeration value="tcp-socket"/>
            <xs:enumeration value="udp-socket"/>
            <xs:enumeration value="file"/>
            <xs:enumeration value="other"/>
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
<xs:attribute name="pathInfo" type="xs:string"
    use="optional"/>
<xs:attribute name="certificate" type="xs:string"
    use="optional"/>
</xs:complexType>
</xs:element>
```

example:

```
<endpoint
    id="NIPRWebService"
    protocol="https"
    ipAddress="100.101.102.103"
    sendPort="8080"
    type="server"
    pathInfo="/service/WebService"
    certificate="certificates/sipr-cert.pki">
    <description>Web service on the NIPR network</description>
</endpoint>
```

### 5.3.8 Domains Example

Cumulative example, including **Data Flow Metadata**, **Guards**, and **Domains**:

```
<?xml version="1.0" encoding="UTF-8" ?>
<dfc
    xmlns:IC="urn:us:gov:ic:ism:v2"
    xmlns:ddms="http://metadata.dod.mil/mdr/ns/DDMS/1.3/"
    xmlns="http://cie.jfcom.mil/schema/dfc"
    version="1.2.11">

    <name>Biometrics</name>
    <uuid>75aa38db-9a0b-4cb4-8d0a-f1e95fc761e7</uuid>
    <version>1.0</version>
```

**UNCLASSIFIED**

```
<created>2008-01-01</created>
<createdBy>snappc</createdBy>
<modified>2008-01-04</modified>
<modifiedBy>duncans</modifiedBy>
<description>This is the DFC for Biometrics demo</description>
<ddms:security IC:ownerProducer="USA" IC:classification="U" />
<expirationDate>2010-01-01</expirationDate>
<enableDate>2008-01-01</enableDate>
<ticketRequest>2008-01-01</ticketRequest>

<guard name="WSG" version="1.0" />
<guard name="DSG" version="3.0">
    <xformConfig xforms="guardconfig/dsg-3.0.xhtml"
        css="guardconfig/dsg-3.0.css" instance="guardconfig/dsg-
        3.0-instance.xml" />
</guard>

<domains>
    <domain id="nipr">
        <description>This is the NIPR domain</description>
        <allowedInboundClassifications>
            <ddms:security
                IC:ownerProducer="USA"
                IC:classification="U" />
        </allowedInboundClassifications>
        <endpoints>
            <endpoint
                id="NIPRWebService"
                protocol="https"
                ipAddress="100.101.102.103"
                sendPort="8080"
                type="server"
                pathInfo="/service/WebService"
                certificate="certificates/sipr-cert.pki">
                <description>Web service on the NIPR
                    network</description>
            </endpoint>
        </endpoints>
    </domain>
    <domain id="sipr">
        <description>This is the SIPR domain</description>
        <minimumAllowedInboundClassifications>
            <ddms:security IC:ownerProducer="USA"
                IC:classification="U" />
        </minimumAllowedInboundClassifications>
        <maximumAllowedInboundClassifications>
            <ddms:security IC:ownerProducer="USA"
                IC:classification="S" />
        </maximumAllowedInboundClassifications>
        <endpoints>
            <endpoint
                id="SIPRWebService"
                protocol="https"
                ipAddress="200.201.202.203"
                sendPort="8080"
                type="server"
                pathInfo="/service/WebService" />
        </endpoints>
    </domain>
</domains>
```

## UNCLASSIFIED

```
    </endpoints>
  </domain>
</domains>
</dfc>
```

### 5.4 Files

#### 5.4.1 <files> element

This element is a list of all the **Schemas**, **Schematrons**, **Stylesheets**, and **Other** files in the **Data Flow**. This list does not contain the subfiles included by these files.

schema:

```
<xs:element name="files" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:sequence>
      [See 5.4.2 for definition of schemas element]
      [See 5.4.4 for definition of schematrons element]
      [See 5.4.6 for definition of stylesheets element]
      [See 5.4.8 for definition of others element]
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

example:

```
<files>
  <schemas>
    <schema id="Event Or Blog" type="W3C"
      file="schemas/Event Or Blog/addEventOrBlog.xsd">
      <description>This schema is for event or blog
      objects.</description>
    </schema>
    <schema id="Correlation Id" type="W3C"
      file="schemas/correlationId.xsd" />
  </schemas>

  <schematrons>
    <schematron id="example" file="schematrons/
      example/example.strn">
      <description>This schematron is for event or blog
      objects.</description>
    </schematron>
    <schematron id="example2" file="schematrons/
      example2/example2.strn" />
  </schematrons>

  <stylesheets>
    <stylesheet id="Event Or Blog"
      file="stylesheets/Event Or Blog/addEventOrBlog.xsl">
      <description>This stylesheet is for event or blog
      objects.</description>
    </stylesheet>
    <stylesheet id="example" file="stylesheets/">
```

## UNCLASSIFIED

```
        example/example.xsl" />
    </stylesheets>
    <others>
        <other id="Test Data" type="Test Data"
            file="others/Test Data/testData.xml" >
            <description>This is an example XML message to use as a
                test</description>
        </other>
    </others>
</files>
```

### 5.4.2 <schemas> element (subelement of: files)

This element is a list of all the **Schemas** in the **Data Flow**. The default path where the **Schemas** are stored is “schemas/”.

schema:

```
<xs:element name="schemas" minOccurs="0" maxOccurs="1">
    <xs:complexType>
        <xs:sequence>
            [See 5.4.3 for definition of schema element]
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

example:

```
<schemas>
    <schema id="Event Or Blog" type="W3C"
        file="schemas/Event Or Blog/addEventOrBlog.xsd">
        <description>This schema is for event or blog
            objects.</description>
    </schema>
    <schema id="Correlation Id" type="W3C"
        file="schemas/Correlation Id/correlationId.xsd" />
</schemas>
```

### 5.4.3 <schema> element (subelement of: schemas)

The element defines a **Schema** file. The default path where the **Schemas** are stored is “schemas/”. The required **id** attribute must be unique within the list of schemas. (See schema from section 5 for the schema xs:key definition.) The required **type** attribute defines the Schema type. Allowed values are “W3C” (W3C XML Schema) and “RelaxNG” (RELAX NG.) The required **file** attribute contains the full DFC path and name of the **Schema**. There is an optional **<description>** element allowing a text description to be added.

schema:

```
<xs:element name="schema" minOccurs="0" maxOccurs="unbounded">
```

## UNCLASSIFIED

```
<xs:complexType>
  <xs:sequence>
    <xs:element name="description" minOccurs="0"
      maxOccurs="1"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="type" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="W3C"/>
        <xs:enumeration value="RelaxNG"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="file" type="xs:string" use="required"/>
</xs:complexType>
</xs:element>
```

example:

```
<schema id="Event Or Blog" type="W3C"
  file="schemas/Event Or Blog/addEventOrBlog.xsd">
  <description>This schema is for event or blog
  objects.</description>
```

### 5.4.4 <schematrons> element (subelement of: files)

This element is a list of all the **Schematrons** in the **Data Flow**. The default path where the **Schematrons** are stored is “schematrons/”.

schema:

```
<xs:element name="schematrons" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:sequence>
      [See 5.4.5 for definition of schematron element]
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

example:

```
<schematrons>
  <schematron id="example"
    file="schematrons/example/example.strn">
    <description>This schematron is for event or blog
    objects.</description>
  </schematron>

  <schematron id="example2" file="schematrons/
    example2/example2.strn" />
</schematrons>
```

### 5.4.5 <schematron> element (subelement of: schematrons)

## UNCLASSIFIED

This element defines a **Schematron** file. The default path where the **Schematrons** are stored is “schematrons/”. The required `id` attribute must be unique within the list of **Schematrons**. (See schema from section 5 for the schema `xs:key` definition.) The required `file` attribute contains the full DFC path and name of the **Schematron**. There is an optional `<description>` element allowing a text description to be added.

schema:

```
<xs:element name="schematron" minOccurs="0"
    maxOccurs="unbounded">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="description" minOccurs="0"
                maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:string" use="required"/>
        <xs:attribute name="file" type="xs:string" use="required"/>
    </xs:complexType>
</xs:element>
```

example:

```
<schematron id="example"
    file="schematrons/example/example.strn">
    <description>This schematron is for event or blog
    objects.</description>
</schematron>
```

### 5.4.6 `<stylesheets>` element (subelement of: `files`)

This element is a list of all the **Stylesheets** in the **Data Flow**. The default path where the **Stylesheets** are stored is “stylesheets/”.

schema:

```
<xs:element name="stylesheets" minOccurs="0" maxOccurs="1">
    <xs:complexType>
        <xs:sequence>
            [See 5.4.7 for definition of stylesheet element]
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

example:

```
<stylesheets>
    <stylesheet id="Event Or Blog"
        file="stylesheets/Event Or Blog/addEventOrBlog.xsl">
        <description>This stylesheet is for event or blog
        objects.</description>
    </stylesheet>

    <stylesheet id="example" file="stylesheets/
        example/example.xsl" />
</stylesheets>
```

## UNCLASSIFIED

### 5.4.7 <stylesheet> element (subelement of: stylesheets)

This element defines a **Stylesheet** file. The default path where the **Stylesheets** are stored is “stylesheets/”. The required `id` attribute must be unique within the list of Stylesheets. (See schema from section 5 for the schema xs:key definition.) The required `file` attribute contains the full DFC path and name of the **Stylesheet**. There is an optional `<description>` element allowing a text description to be added.

schema:

```
<xs:element name="stylesheet" minOccurs="0"
    maxOccurs="unbounded">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="description" minOccurs="0"
                maxOccurs="1"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:string" use="required"/>
        <xs:attribute name="file" type="xs:string" use="required"/>
    </xs:complexType>
</xs:element>
```

example:

```
<stylesheet id="Event Or Blog"
    file="stylesheets/Event Or Blog/addEventOrBlog.xsl">
    <description>This stylesheet is for event or blog
        objects.</description>
</stylesheet>
```

### 5.4.8 <others> element (subelement of: files)

This element is a list of files that do not fit into the categories of Schema, Schematron, or Stylesheet. The default path where these files are stored is “others/”.

schema:

```
<xs:element name="others" minOccurs="0" maxOccurs="1">
    <xs:complexType>
        <xs:sequence>
            [See 5.4.9 for definition of other element]
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

example:

```
<others>
    <other id="Test Data" type="Test Data"
        file="others/Test Data/testData.xml" >
        <description>This is an example XML message to use as a
            test</description>
    </other>
</others>
```

## UNCLASSIFIED

### 5.4.9 <other> element (subelement of: others)

This element defines a file that does not fit into the categories of Schema, Schematron, or Stylesheet. The default path where these files are stored is “others/”. The required `id` attribute must be unique within the list of files. (See schema from section 5 for the other xs:key definition.) The required `file` attribute contains the full DFC path and name of the file. The required `type` attribute provides a means to specify the file type. For test data, this value should be “Test Data”. There is an optional `<description>` element allowing a text description to be added.

schema:

```
<xs:element name="other" minOccurs="0"
    maxOccurs="unbounded">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="description" minOccurs="0"
                maxOccurs="1" />
        </xs:sequence>
        <xs:attribute name="id" type="xs:string" use="required"/>
        <xs:attribute name="file" type="xs:string" use="required"/>
        <xs:attribute name="type" type="xs:string" use="optional"/>
    </xs:complexType>
</xs:element>
```

example:

```
<other id="Test Data" type="Test Data"
    file="others/Test Data/testData.xml" >
    <description>This is an example XML message to use as a
    test</description>
</other>
```

### 5.4.10 Files Example

Cumulative example, including **Data Flow Metadata, Guards, Domains** and **Files**:

```
<?xml version="1.0" encoding="UTF-8" ?>
<dfc
    xmlns:IC="urn:us:gov:ic:ism:v2"
    xmlns:ddms="http://metadata.dod.mil/mdr/ns/DDMS/1.3/"
    xmlns="http://cie.jfcom.mil/schema/dfc"
    version="1.2.11">

    <name>Biometrics</name>
    <uuid>75aa38db-9a0b-4cb4-8d0a-f1e95fc761e7</uuid>
    <version>1.0</version>
    <created>2008-01-01</created>
    <createdBy>snappc</createdBy>
    <modified>2008-01-04</modified>
    <modifiedBy>duncans</modifiedBy>
    <description>This is the DFC for Biometrics demo</description>
    <ddms:security IC:ownerProducer="USA" IC:classification="U" />
    <expirationDate>2010-01-01</expirationDate>
```

**UNCLASSIFIED**

```
<enableDate>2008-01-01</enableDate>
<ticketRequest>2008-01-01</ticketRequest>

<guard name="WSG" version="1.0" />
<guard name="DSG" version="3.0">
  <xformConfig xforms="guardconfig/dsg-3.0.xhtml"
    css="guardconfig/dsg-3.0.css" instance="guardconfig/dsg-
    3.0-instance.xml" />
</guard>

<domains>
  <domain id="nipr">
    <description>This is the NIPR domain</description>
    <allowedInboundClassifications>
      <ddms:security
        IC:ownerProducer="USA"
        IC:classification="U" />
    </allowedInboundClassifications>
    <endpoints>
      <endpoint
        id="NIPRWebService"
        protocol="https"
        ipAddress="100.101.102.103"
        sendPort="8080"
        type="server"
        pathInfo="/service/WebService"
        certificate="certificates/sipr-cert.pki">
        <description>Web service on the NIPR
          network</description>
      </endpoint>
    </endpoints>
  </domain>
  <domain id="sipr">
    <description>This is the SIPR domain</description>
    <minimumAllowedInboundClassifications>
      <ddms:security IC:ownerProducer="USA"
        IC:classification="U" />
    </minimumAllowedInboundClassifications>
    <maximumAllowedInboundClassifications>
      <ddms:security IC:ownerProducer="USA"
        IC:classification="S" />
    </maximumAllowedInboundClassifications>
    <endpoints>
      <endpoint
        id="SIPRWebService"
        protocol="https"
        ipAddress="200.201.202.203"
        sendPort="8080"
        type="server"
        pathInfo="/service/WebService" />
    </endpoints>
  </domain>
</domains>

<files>
  <schemas>
    <schema id="Event Or Blog" type="W3C"
```

## UNCLASSIFIED

```
        file="schemas/Event Or Blog/addEventOrBlog.xsd">
        <description>This schema is for event or blog
        objects.</description>
    </schema>
    <schema id="Correlation Id" type="W3C"
        file="schemas/Correlation Id/correlationId.xsd" />
</schemas>
<schematrons>
    <schematron id="example" file="schematrons/
        example/example.strn">
        <description>This schematron is for event or blog
        objects.</description>
    </schematron>
    <schematron id="example2" file="schematrons/
        example2/example2.strn" />
</schematrons>
<stylesheets>
    <stylesheet id="Event Or Blog"
        file="stylesheets/Event Or Blog/addEventOrBlog.xsl">
        <description>This stylesheet is for event or blog
        objects.</description>
    </stylesheet>
    <stylesheet id="example" file="stylesheets/
        example/example.xsl" />
</stylesheets>
<others>
    <other id="Test Data" type="Test Data"
        file="others/Test Data/testData.xml" >
        <description>This is an example XML message to use as a
        test</description>
    </other>
</others>
</files>
</dfc>
```

## 5.5 Domain Pairs

### 5.5.1 <*domainPairs*> element

This element is the list of **Domain Pairs** for the **Data Flow**. The default directory for storing files associated with the Domain Pair is “domainPairs/”.

schema:

```
<xs:element name="domainPairs" minOccurs="1" maxOccurs="1">
    <xs:complexType>
        <xs:sequence>
            [ See 5.5.2 for definition of domainPair element]
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

example:

```
<domainPairs>
```

**UNCLASSIFIED**

```
<domainPair domain1Ref="sipr" domain2Ref="nipr">
  <filterset id="dsg-filters1" fromDomainRef="sipr"
    toDomainRef="nipr">
    <guard name="DSG" version="3.0">
      <options>
        <option name="max_filter_memory_size">1000000</option>
      </options>
      <xformConfig
        xforms="guardconfig/dsg-3.0-filterset.xhtml"
        css="guardconfig/dsg-3.0-filterset.css"
        instance="domainPairs/sipr_nipr/dsg-filters1
          /guardconfig/dsg-3.0-instance.xml" />
    </guard>

    <fromEndpoints>
      <endpoint ref="SIPRWebService" />
    </fromEndpoints>

    <toEndpoints>
      <endpoint ref="NIPRWebService" />
    </toEndpoints>

    <requires type="all">
      <require filtersetRef="dsg-filters2" />
    </requires>

    <associates>
      <associate filtersetRef="dsg-filters2"/>
    </associates>

    <filters>
      <xmlvalidator id="1" schemaRef="Correlation Id">
        <filterDataInput location="original"/>
      </xmlvalidator>
    </filters>
  </filterset>

  <filterset id="dsg-filters2" fromDomainRef="sipr"
    toDomainRef="nipr">
    <guard name="DSG" version="3.0">
      <options>
        <option name="max_filter_memory_size">1000000</option>
      </options>

      <xformConfig
        xforms="guardconfig/dsg-3.0-filterset.xhtml"
        css="guardconfig/dsg-3.0-filterset.css"
        instance="domainPairs/sipr_nipr/dsg-filters2
          /guardconfig/dsg-3.0-instance.xml" />
    </guard>

    <fromEndpoints>
      <endpoint ref="SIPRWebService" />
    </fromEndpoints>

    <toEndpoints>
```

**UNCLASSIFIED**

```
<endpoint ref="NIPRWebService" />
</toEndpoints>

<requires type="all">
  <require filtersetRef="dsg-filters1" />
</requires>

<dependencies>
  <dependency filtersetRef="dsg-filters1" />
</dependencies>

<associates>
  <associate filtersetRef="dsg-filters1"/>
</associates>

<filters>
  <xmlvalidator id="1" schemaRef="Event or Blog">
    <filterDataInput location="original" />
  </xmlvalidator>

  <wordchecker id="2" file="domainPairs/sipr_nipr/dsg-
    filters2/2/dirty-clean-words.xml">
    <dependencies>
      <dependency filterRef="1" />
    </dependencies>
    <filterDataInput location="dependency"
      dependencyFilterRef="1" />
  </wordchecker>

  <xmlnormalizer id="3">
    <dependencies>
      <dependency filterRef="2" />
    </dependencies>
    <filterDataInput location="dependency"
      dependencyFilterRef="2" />
  </xmlnormalizer>

  <classificationchecker id="4">
    <dependencies>
      <dependency filterRef="3" />
    </dependencies>
    <filterDataInput location="dependency"
      dependencyFilterRef="3" />
  </classificationchecker>

  <viruschecker id="5">
    <dependencies>
      <dependency filterRef="4" />
    </dependencies>
    <filterDataInput location="original" />
  </viruschecker>

  <xsltransformer id="6" stylesheetRef="Event Or Blog">
    <dependencies>
      <dependency filterRef="5" />
    </dependencies>
  </xsltransformer>

```

**UNCLASSIFIED**

```
<filterDataInput location="dependency"
    dependencyFilterRef="5" />
</xsltransformer>

<xslvalidator id="7" stylesheetRef="Event Or Blog">
    <dependencies>
        <dependency filterRef="6" />
    </dependencies>
    <filterDataInput location="dependency"
        dependencyFilterRef="6" />
</xslvalidator>

<schematron id="8" schematronRef="example">
    <dependencies>
        <dependency filterRef="7" />
    </dependencies>
    <filterDataInput location="dependency"
        dependencyFilterRef="7" />
</schematron>

<digidataidsignervalidator id="9" source="external">
    <urls>
        <url value="OCSP://myca.dod.mil"/>
    </urls>

    <dependencies>
        <dependency filterRef="8" />
    </dependencies>
    <filterDataInput location="dependency"
        dependencyFilterRef="8" />
</digidataidsignervalidator>

<digidataidsignerremover id="10">
    <dependencies>
        <dependency filterRef="9" />
    </dependencies>
    <filterDataInput location="dependency"
        dependencyFilterRef="9" />
</digidataidsignerremover>

<custom id="11" type="CustomParser">
    <description>Custom MagParser filter.</description>
    <files>
        <file id="event" name="event" type="txt"
            file="domainPairs/sipr_nipr/dsg-filters2/11/
                customParser.dat">
            <description>CustomParser for event
                data</description>
        </file>
    </files>
    <options>
        <option name="stripHeader">TRUE</option>
    </options>
    <dependencies>
        <dependency filterRef="10" />
    </dependencies>
    <filterDataInput location="dependency"
```

## UNCLASSIFIED

```
        dependencyFilterRef="10" />
    </custom>
</filters>
</filterset>
</domainPairs>
</domainPairs>
```

### 5.5.2 <domainPair> element (subelement of: domainPairs)

A **Domain Pair** is defined as the association of two domains and the sets of filters that are to be applied to messages being sent between them. The required `domain1Ref` and `domain2Ref` attributes reference the Domains by their id. (See schema of `dfc.xml` for constraints of values for `domain1Ref` and `domain2Ref`) The element contains one or more **Filtersets**. There is an optional `<description>` element allowing a text description to be added.

By default, all files relating to this **Domain Pair** are contained in a directory in the domain pairs directory ("domainPairs/") based on the **Domain Pair's** two domains. ("domainPairs/nipr\_sipr").

schema:

```
<xs:element name="domainPair" minOccurs="1"
maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="description" type="xs:string"
minOccurs="0" maxOccurs="1" />
[See 5.5.3 for definition of filterset element]
</xs:sequence>
<xs:attribute name="domain1Ref" type="xs:string"
use="required" />
<xs:attribute name="domain2Ref" type="xs:string"
use="required" />
</xs:complexType>

<xs:key name="filtersetKey">
<xs:selector xpath="dfc:filterset" />
<xs:field xpath="@id" />
</xs:key>

<xs:keyref name="requiresKeyref" refer="dfc:filtersetKey">
<xs:selector xpath="dfc:filterset/dfc:requires/
dfc:require" />
<xs:field xpath="@filtersetRef" />
</xs:keyref>

<xs:keyref name="dependencyKeyref" refer="dfc:filtersetKey">
<xs:selector xpath="dfc:filterset/dfc:dependencies/
dfc:dependency" />
<xs:field xpath="@filtersetRef" />
</xs:keyref>

<xs:keyref name="associateKeyref" refer="dfc:filtersetKey">
<xs:selector
```

## UNCLASSIFIED

```
    xpath="dfc:filterset/dfc:associates/dfc:associate" />
<xs:field xpath="@filtersetRef" />
</xs:keyref>
</xs:element>
```

example:

```
<domainPair domain1Ref="sipr" domain2Ref="nipp">
  <filterset id="dsg-filters1" fromDomainRef="sipr"
    toDomainRef="nipp">
    <guard name="DSG" version="3.0">
      <options>
        <option name="max_filter_memory_size">1000000</option>
      </options>

      <xformConfig
        xforms="guardconfig/dsg-3.0-filterset.xhtml"
        css="guardconfig/dsg-3.0-filterset.css"
        instance="domainPairs/sipr_nipp/dsg-filters1
          /guardconfig/dsg-3.0-instance.xml" />
    </guard>

    <fromEndpoints>
      <endpoint ref="SIPRWebService" />
    </fromEndpoints>

    <toEndpoints>
      <endpoint ref="NIPRWebService" />
    </toEndpoints>

    <requires type="all">
      <require filtersetRef="dsg-filters2" />
    </requires>

    <associates>
      <associate filtersetRef="dsg-filters2" />
    </associates>

    <filters>
      <xmlvalidator id="1" schemaRef="Correlation Id">
        <filterDataInput location="original" />
      </xmlvalidator>
    </filters>
  </filterset>

  <filterset id="dsg-filters2" fromDomainRef="sipr"
    toDomainRef="nipp">
    <guard name="DSG" version="3.0">
      <options>
        <option name="max_filter_memory_size">1000000</option>
      </options>
      <xformConfig
        xforms="guardconfig/dsg-3.0-filterset.xhtml"
        css="guardconfig/dsg-3.0-filterset.css"
        instance="domainPairs/sipr_nipp/dsg-filters2
          /guardconfig/dsg-3.0-instance.xml" />
    </guard>
```

**UNCLASSIFIED**

```
</guard>

<fromEndpoints>
  <endpoint ref="SIPRWebService" />
</fromEndpoints>

<toEndpoints>
  <endpoint ref="NIPRWebService" />
</toEndpoints>

<requires type="all">
  <require filtersetRef="dsg-filters1" />
</requires>

<dependencies>
  <dependency filtersetRef="dsg-filters1" />
</dependencies>

<associates>
  <associate filtersetRef="dsg-filters1"/>
</associates>

<filters>
  <xmvalidator id="1" schemaRef="Event or Blog">
    <filterDataInput location="original" />
  </xmvalidator>

  <wordchecker id="2" file="domainPairs/sipr_nipr/dsg-
    filters2/2/dirty-clean-words.xml">
    <dependencies>
      <dependency filterRef="1" />
    </dependencies>
    <filterDataInput location="dependency"
      dependencyFilterRef="1" />
  </wordchecker>

  <xmnormalizer id="3">
    <dependencies>
      <dependency filterRef="2" />
    </dependencies>
    <filterDataInput location="dependency"
      dependencyFilterRef="2" />
  </xmnormalizer>

  <classificationchecker id="4">
    <dependencies>
      <dependency filterRef="3" />
    </dependencies>
    <filterDataInput location="dependency"
      dependencyFilterRef="3" />
  </classificationchecker>

  <viruschecker id="5">
    <dependencies>
      <dependency filterRef="4" />
    </dependencies>
    <filterDataInput location="original" />
  </viruschecker>
</filters>
```

## UNCLASSIFIED

```
</viruschecker>

<xsltransformer id="6" stylesheetRef="Event Or Blog">
    <dependencies>
        <dependency filterRef="5" />
    </dependencies>
    <filterDataInput location="dependency"
        dependencyFilterRef="5" />
</xsltransformer>

<xslvalidator id="7" stylesheetRef="Event Or Blog">
    <dependencies>
        <dependency filterRef="6" />
    </dependencies>
    <filterDataInput location="dependency"
        dependencyFilterRef="6" />
</xslvalidator>

<schematron id="8" schematronRef="example">
    <dependencies>
        <dependency filterRef="7" />
    </dependencies>
    <filterDataInput location="dependency"
        dependencyFilterRef="7" />
</schematron>

<digtalsignaturevalidator id="9" source="external">
    <urls>
        <url value="OCSP://myca.dod.mil"/>
    </urls>

    <dependencies>
        <dependency filterRef="8" />
    </dependencies>
    <filterDataInput location="dependency"
        dependencyFilterRef="8" />
</digtalsignaturevalidator>

<digtalsignatureremover id="10">
    <dependencies>
        <dependency filterRef="9" />
    </dependencies>
    <filterDataInput location="dependency"
        dependencyFilterRef="9" />
</digtalsignatureremover>

<custom id="11" type="CustomParser">
    <description>Custom MagParser filter.</description>
    <files>
        <file id="event" name="event" type="txt"
            file="domainPairs/sipr_nipr/dsg-filters2/11/
            customParser.dat">
            <description>CustomParser for event
                data</description>
        </file>
    </files>
    <options>
```

## UNCLASSIFIED

```
        <option name="stripHeader">TRUE</option>
    </options>
    <dependencies>
        <dependency filterRef="11" />
    </dependencies>
    <filterDataInput location="dependency"
        dependencyFilterRef="10" />
    </custom>
</filters>
</filterset>
</domainPair>
```

### 5.5.3 *<filterset> element (subelement of: domainPair)*

This element contains the list of **Filtersets** for **Domain Pair**. The required `id` attribute must be unique within the list of **Filtersets** for the **Domain Pair**. The `id` must be alphanumeric, dashes (-) and underscores (\_). The `id` must be 64 characters or less in length. (See schema for `<domainPair>` for the schema `xs:key` definition.) Multiple filtersets allow for several use cases to be satisfied for guards. The first use case is to allow multiple messages to be sent for a single Data Flow. When there is more than one message in a Data Flow, the filterset `id`'s should be named according to the message's `id` (or segment name). For example, the DataSyncGuard uses the `id` attribute as its message type to determine which filter rules to apply to the message, and the WSG uses the `id` attribute to determine which WSG pipeline (inbound-request, inbound-response, outbound-response, and outbound-request) is being configured.

The required `fromDomainRef` and `toDomainRef` attributes are used to specify the direction of flow in the **Domain Pair** for the **Filterset**. (See schema of `dfc.xml` for constraints of values for `fromDomainRef` and `toDomainRef`)

There is an optional `<description>` element allowing a text description to be added. It is recommended that the `description` element be used to describe why certain filters were chosen, how they are validating the data, and why the specific order was chosen. Additionally, users may find it helpful to use the filterset description to describe how the filters are being used to mitigate any medium and high risk findings from the Schema lockdown report, such as using a Stylesheet to transform some data that cannot be adequately constrained in schema.

The element contains a **Guard**, **From Endpoints**, **To Endpoints**, **Required Filterset**, **Dependency Filtersets**, **Associated Filtersets**, and a list of **Filters**.

By default, all files relating to this **Filterset** are contained in a directory in the **Domain Pair** directory named by the **Filterset** id. ("domainPairs/sipr\_nipr/outbound-request/.")

schema:

```
<xs:element name="filterset" minOccurs="1" maxOccurs="unbounded">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="description" minOccurs="0"
                maxOccurs="1" />
            [See 5.5.4 for definition of guard element]
```

## UNCLASSIFIED

```
[See 5.5.8 for definition of fromEndpoints element]
[See 5.5.10 for definition of toEndpoints element]
[See 5.5.12 for definition of requires element]
[See 5.5.14 for definition of dependencies element]
[See 5.5.16 for definition of associates element]
[See 5.5.18 for definition of filters element]
</xs:sequence>
<xs:attribute name="id" type="xs:string" use="required" />
<xs:attribute name="fromDomainRef" type="xs:string"
    use="required" />
<xs:attribute name="toDomainRef" type="xs:string"
    use="required" />
</xs:complexType>
<xs:key name="filtersKey">
    <xs:selector xpath="dfc:filters/*" />
    <xs:field xpath="@id" />
</xs:key>
<xs:keyref name="filtersDependencyKeyref"
    refer="dfc:filtersKey">
    <xs:selector
        xpath="dfc:filters/*/dfc:dependencies/dfc:dependency" />
        <xs:field xpath="@filterRef" />
</xs:keyref>
<xs:keyref name="filtersDataInputDependencyKeyref"
    refer="dfc:filtersKey">
    <xs:selector xpath="dfc:filters/*/dfc:filterDataInput" />
    <xs:field xpath="@dependencyFilterRef" />
</xs:keyref>
</xs:element>
```

example:

```
<filterset id="dsg-filters2" fromDomainRef="sipr"
    toDomainRef="nipr">
    <guard name="DSG" version="3.0">
        <options>
            <option name="max_filter_memory_size">1000000</option>
        </options>
        <xformConfig
            xforms="guardconfig/dsg-3.0-filterset.xhtml"
            css="guardconfig/dsg-3.0-filterset.css"
            instance="domainPairs/sipr_nipr/dsg-filters2/guardconfig/
            dsg-3.0-instance.xml" />
    </guard>

    <fromEndpoints>
        <endpoint ref="SIPRWebService" />
    </fromEndpoints>

    <toEndpoints>
        <endpoint ref="NIPRWebService" />
    </toEndpoints>

    <requires type="all">
        <require filtersetRef="dsg-filters1" />
    </requires>
```

## UNCLASSIFIED

```
<dependencies>
  <dependency filtersetRef="dsg-filters1" />
</dependencies>

<associates>
  <associate filtersetRef="dsg-filters2"/>
</associates>

<filters>
  <xmlvalidator id="1" schemaRef="Event or Blog">
    <filterDataInput location="original"/>
  </xmlvalidator>

  <wordchecker id="2" file="domainPairs/sipr_nipr/dsg-filters2
  /2/dirty-clean-words.xml">
    <dependencies>
      <dependency filterRef="1" />
    </dependencies>
    <filterDataInput location="dependency"
      dependencyFilterRef="1" />
  </wordchecker>

  <xmlnormalizer id="3">
    <dependencies>
      <dependency filterRef="2" />
    </dependencies>
    <filterDataInput location="dependency"
      dependencyFilterRef="2" />
  </xmlnormalizer>

  <classificationchecker id="4">
    <dependencies>
      <dependency filterRef="3" />
    </dependencies>
    <filterDataInput location="dependency"
      dependencyFilterRef="3" />
  </classificationchecker>

  <viruschecker id="5">
    <dependencies>
      <dependency filterRef="4" />
    </dependencies>
    <filterDataInput location="original"/>
  </viruschecker>

  <xsltransformer id="6" stylesheetRef="Event Or Blog">
    <dependencies>
      <dependency filterRef="5" />
    </dependencies>
    <filterDataInput location="dependency"
      dependencyFilterRef="5" />
  </xsltransformer>

  <xslvalidator id="7" stylesheetRef="Event Or Blog">
    <dependencies>
```

**UNCLASSIFIED**

```
<dependency filterRef="6" />
</dependencies>
<filterDataInput location="dependency"
    dependencyFilterRef="6" />
</xslvalidator>

<schematron id="8" schematronRef="example">
    <dependencies>
        <dependency filterRef="7" />
    </dependencies>
    <filterDataInput location="dependency"
        dependencyFilterRef="7" />
</schematron>

<digtalsignaturevalidator id="9" source="external">
    <urls>
        <url value="OCSP://myca.dod.mil"/>
    </urls>

    <dependencies>
        <dependency filterRef="8" />
    </dependencies>
    <filterDataInput location="dependency"
        dependencyFilterRef="8" />
</digtalsignaturevalidator>

<digtalsignatureremover id="10">
    <dependencies>
        <dependency filterRef="9" />
    </dependencies>
    <filterDataInput location="dependency"
        dependencyFilterRef="9" />
</digtalsignatureremover>

<custom id="11" type="CustomParser">
    <description>Custom MagParser filter.</description>
    <files>
        <file id="event" name="event" type="txt"
            file="domainPairs/sipr_nipr/dsg-filters2/11/
                customParser.dat">
            <description>CustomParser for event data</description>
        </file>
    </files>
    <options>
        <option name="stripHeader">TRUE</option>
    </options>
    <dependencies>
        <dependency filterRef="10" />
    </dependencies>
    <filterDataInput location="dependency"
        dependencyFilterRef="10" />
</custom>
</filters>
</filterset>
```

## UNCLASSIFIED

### 5.5.4 <guard> element (subelement of: filterset)

This optional element associates a single **Guard** that this **Filterset** will be used with. The required `name` and `version` attributes define the guard's name and version. Each guard can have an optional **XForms Configuration** for the **Guard** to configure guard-specific properties at the Filterset level.

There is a `generic` guard value that is used when a DFCF author does not want to limit the filterset to a specific guard's features/capabilities.

The optional `messageType` attribute used by the guard to determine which filterset is being used for the incoming messages. It is the same as the segment type.

The optional `nativeSupport` attribute is a boolean value defining whether the guard supports DFC natively.

schema:

```
<xs:element name="guard" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:sequence>
      [See 5.5.5 for definition of options element]
      [See 5.5.7 for definition of xformsConfig element]
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required"/>
    <xs:attribute name="version" type="xs:string"
      use="required"/>
    <xs:attribute name="messageType" type="xs:string"
      use="optional"/>
    <xs:attribute name="nativeSupport" type="xs:boolean"
      use="optional"/>
  </xs:complexType>
  <xs:key name="filtersetGuardOptionsKey">
    <xs:selector xpath="dfc:options/dfc:option" />
    <xs:field xpath="@name" />
  </xs:key>
</xs:element>
```

example:

```
<guard name="DSG" version="3.0">
  <options>
    <option name="max_filter_memory_size">1000000</option>
  </options>
  <xformConfig xforms="guardconfig/dsg-3.0-filterset.xhtml"
    css="guardconfig/dsg-3.0-filterset.css"
    instance="domainPairs/sipr_nipr/dsg-filters/guardconfig/dsg-
    3.0-instance.xml" />
</guard>
```

### 5.5.5 <options> element (subelement of: guard)

## UNCLASSIFIED

This optional element provides a way to configure simple guard-specific information without the overhead of an XForms form being defined.

schema:

```
<xs:complexType name="options">
  <xs:sequence>
    [See 5.5.6 definition of option element]
  </xs:sequence>
</xs:complexType>
```

example:

```
<options>
  <option name="max filter memory size">10000000</option>
</options>
```

### 5.5.6 *<option> element (subelement of: options)*

This element contains one or more name/value pairs. The required `name` attribute must be unique within the scope of the **Guard**. The text element of `<option>` contains the value, allowing Unicode characters to be included in the value.

schema:

```
<xs:element name="option" minOccurs="1" maxOccurs="unbounded">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="name" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

example:

```
<option name="max filter memory size">10000000</option>
```

### 5.5.7 *<xformConfig> element (subelement of: guard)*

This optional element associates a **Xforms Configuration** with a **Filterset's Guard** element. The required `xforms` attribute contains the path to the XForms form file. The optional `css` attribute contains the path to the stylesheet to decorate the XForms form. The required `instance` attribute contains the path to the guard properties file that is generated after the user completes the XForms form.

DFCF 1.2.11-based XForms must be compliant with version 1.0 of World Wide Web Consortium's XForms specification (<http://www.w3.org/MarkUp/Forms>).

By default, the XForms form file and stylesheet are in the main Guard directory. The instance file is in a directory in the Filterset directory named "guardconfig".  
("domainPairs/nipr\_sipr/dsg-filters/guardconfig/")

schema:

## UNCLASSIFIED

```
<xs:element name="xformConfig" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:attribute name="xforms" type="xs:string" use="required"/>
    <xs:attribute name="css" type="xs:string" use="optional"/>
    <xs:attribute name="instance" type="xs:string"
      use="required"/>
  </xs:complexType>
</xs:element>
```

example:

```
<xformConfig xforms="/guardconfig/dsg-3.0-filterset.xhtml"
  css="/guardconfig/dsg-3.0-filterset.css"
  instance="/domainPairs/sipr_nipr/dsg-filters/guardconfig/dsg-
  3.0-instance.xml" />
```

### 5.5.8 <fromEndpoints> element (subelement of: filterset)

The optional <fromEndpoints> element provides a way to specify the **Endpoints** originating the message this **Filterset** will use.

schema:

```
<xs:element name="fromEndpoints" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:sequence>
      [See 5.5.9 for definition of endpoint element]
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

example:

```
<fromEndpoints>
  <endpoint ref="SIPRWebService" />
</fromEndpoints>
```

### 5.5.9 <endpoint> element (subelement of: fromEndpoints)

This element lists a single **Endpoint** that originate the message this **Filterset** will use. The required `ref` attribute references an **Endpoint** by its `id`. The referenced **Endpoint** must exist in the **Domain Pair's** referenced **Domains**, and must not be in the same **Domain** as the `toEndpoint`. (See schema of dfc.xml for constraints of values for `ref`)

schema:

```
<xs:element name="endpoint" minOccurs="1" maxOccurs="1">
  <xs:complexType>
    <xs:attribute name="ref" type="xs:string" />
  </xs:complexType>
</xs:element>
```

## UNCLASSIFIED

example:

```
<endpoint ref="SIPRWebService" />
```

### 5.5.10 <toEndpoint> element (subelement of: filterset)

The optional <toEndpoints> element provides a way to specify the **Endpoints** receiving the message this **Filterset** will use.

schema:

```
<xs:element name="toEndpoints" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:sequence>
      [See 5.5.11 for definition of endpoint element]
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

example:

```
<toEndpoints>
  <endpoint ref="NIPRWebService" />
</toEndpoints>
```

### 5.5.11 <endpoint> element (subelement of: toEndpoints)

This element lists a single **Endpoint** that receives the message this **Filterset** will use. The required `ref` attribute references an **Endpoint** by its `id`. The referenced **Endpoint** must exist in the **Domain Pair's** referenced **Domains**, and must not be in the same **Domain** as the fromEndpoint. (See schema of dfc.xml for constraints of values for `ref`)

schema:

```
<xs:element name="endpoint" minOccurs="1" maxOccurs="1">
  <xs:complexType>
    <xs:attribute name="ref" type="xs:string" />
  </xs:complexType>
</xs:element>
```

example:

```
<endpoint ref="NIPRWebService" />
```

### 5.5.12 <requires> element (subelement of: filterset)

The optional <requires> element provides a way for a **Filterset** to support message segmentation. It has a list of references to other **Filtersets** that are required to be successfully processed before this **Filterset** is considered completed. For example, a **Filterset** for an image may require a **Filterset** for an associated XML file containing metadata describing the image. This metadata filterset must be processed along with the image filterset in order for either of them to be considered complete. This element does not specify an order of execution; instead, it specifies a necessity for the required

## UNCLASSIFIED

**Filtersets** to have been processed successfully for this **Filterset** to be valid. Ordering of filtersets is covered in <dependencies> (sections 5.5.14/5.5.15).

The required `type` attribute can have values of `all` or `any`. A value of `all` means that all required **Filtersets** must succeed, and `any` means that only one of the required **Filterset** must succeed.

schema:

```
<xs:element name="requires" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:sequence>
      [See 5.5.13 for definition of require element]
    </xs:sequence>
    <xs:attribute name="type" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="all"/>
          <xs:enumeration value="any"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
</xs:element>
```

example:

```
<requires type="all">
  <require filtersetRef="dsg-filters1" />
</requires>
```

### 5.5.13 <require> element (subelement of: requires)

One or more references to a **Filterset** id. The required `filterSetRef` attribute must have a valid **Filterset** id.

schema:

```
<xs:element name="require" minOccurs="1" maxOccurs="unbounded">
  <xs:complexType>
    <xs:attribute name="filtersetRef" type="xs:string"
      use="required"/>
  </xs:complexType>
</xs:element>
```

example:

```
<require filtersetRef="dsg-filters1" />
```

### 5.5.14 <dependencies> element (subelement of: filterset)

The optional <dependencies> element provides a way to support **Filterset** ordering (which is necessary for message segmentation.) It has a list of references to other

## UNCLASSIFIED

**Filtersets** that are required to execute and succeed before this **Filterset** may be executed.

schema:

```
<xs:element name="dependencies" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:sequence>
      [See 5.5.15 for definition of dependency element]
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

example:

```
<dependencies>
  <dependency filtersetRef="dsg-filters1" />
</dependencies>
```

### 5.5.15 <dependency> element (subelement of: dependencies)

This element contains a reference to another **Filterset's** id and is used to indicate on which filtersets this filterset depends. More than one <dependency> element can be specified. There is no implied ordering of multiple dependencies; they are considered equal. If dependency ordering is required, then each filterSet should only explicitly state which filterSet it immediately depends on. The required filtersetRef attribute must have a valid **Filterset** id. It must not contain its own id, or the id of any other **Filtersets** that directly or indirectly depend on this **Filterset**, causing circular dependencies.

The dependency only specifies the process order and does not specify how data is passed between filters.

schema:

```
<xs:element name="dependency" minOccurs="1"
  maxOccurs="unbounded">
  <xs:complexType>
    <xs:attribute name="filtersetRef" type="xs:string"
      use="required"/>
  </xs:complexType>
</xs:element>
```

example:

```
<dependency filtersetRef="dsg-filters1" />
```

### 5.5.16 <associates> element (subelement of: filterset)

The optional <associates> element provides a way to associate this **Filterset** with a number of other **Filtersets**.

schema:

```
<xs:element name="associates" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:sequence>
      [See 5.5.17 for definition of associate element]
```

## UNCLASSIFIED

```
</xs:sequence>
</xs:complexType>
</xs:element>
```

example:

```
<associates>
  <associate filtersetRef="dsg-filters1"/>
</associates>
```

### 5.5.17 *<associate> element (subelement of: associates)*

This element contains a reference to another **Filterset's** id and is used to associate **Filterset A** with **Filterset B**. More than one *<associate>* element can be specified. The required filtersetRef attribute must have a valid **Filterset** id. It must not contain its own id. This is typically used to define a request/response relationship between two filtersets.

schema:

```
<xs:element name="associate" minOccurs="1"
  maxOccurs="unbounded">
  <xs:complexType>
    <xs:attribute name="filtersetRef" type="xs:string"
      use="required"/>
  </xs:complexType>
</xs:element>
```

example:

```
<associate filtersetRef="dsg-filters1"/>
```

### 5.5.18 *<filters> element (subelement of: filterset)*

This element is a list of one or more **Filters** for a **Filterset**.

schema:

```
<xs:element name="filters" minOccurs="1" maxOccurs="1">
  <xs:complexType>
    <xs:choice minOccurs="1" maxOccurs="unbounded">
      [See 5.5.19 for definition of xmlvalidator element]
      [See 5.5.20 for definition of schematron element]
      [See 5.5.21 for definition of xsltransformer element]
      [See 5.5.22 for definition of xslvalidator element]
      [See 5.5.23 for definition of wordchecker element]
      [See 5.5.24 for definition of xmlnormalizer element]
      [See 5.5.25 for definition of classificationchecker element]
      [See 5.5.26 for definition of viruschecker element]
      [See 5.5.27 for definition of digitsignaturevalidator
        element]
      [See 5.5.28 for definition of digitsignatureremover
        element]
      [See 5.5.29 for definition of custom element]
    </xs:choice>
```

## UNCLASSIFIED

```
</xs:complexType>  
</xs:element>
```

example:

```
<filters>  
  <xmlvalidator id="1" schemaRef="Event or Blog">  
    <filterDataInput location="original"/>  
  </xmlvalidator>  
  
  <wordchecker id="2" file="domainPairs/sipr_nipr/dsg-filters2  
  /2/dirty-clean-words.xml">  
    <dependencies>  
      <dependency filterRef="1" />  
    </dependencies>  
    <filterDataInput location="dependency"  
      dependencyFilterRef="1" />  
  </wordchecker>  
  
  <xmlnormalizer id="3">  
    <dependencies>  
      <dependency filterRef="2" />  
    </dependencies>  
    <filterDataInput location="dependency"  
      dependencyFilterRef="2" />  
  </xmlnormalizer>  
  
  <classificationchecker id="4">  
    <dependencies>  
      <dependency filterRef="3" />  
    </dependencies>  
    <filterDataInput location="dependency"  
      dependencyFilterRef="3" />  
  </classificationchecker>  
  
  <viruschecker id="5">  
    <dependencies>  
      <dependency filterRef="4" />  
    </dependencies>  
    <filterDataInput location="original"/>  
  </viruschecker>  
  
  <xsltransformer id="6" stylesheetRef="Event Or Blog">  
    <dependencies>  
      <dependency filterRef="5" />  
    </dependencies>  
    <filterDataInput location="dependency"  
      dependencyFilterRef="5" />  
  </xsltransformer>  
  
  <xslvalidator id="7" stylesheetRef="Event Or Blog">  
    <dependencies>  
      <dependency filterRef="6" />  
    </dependencies>  
    <filterDataInput location="dependency"
```

## UNCLASSIFIED

```
        dependencyFilterRef="6" />
    </xslvalidator>

    <schematron id="8" schematronRef="example">
        <dependencies>
            <dependency filterRef="7" />
        </dependencies>
        <filterDataInput location="dependency"
            dependencyFilterRef="7" />
    </schematron>

    <digidatasignaturevalidator id="9" source="external">
        <urls>
            <url value="OCSP://myca.dod.mil"/>
        </urls>

        <dependencies>
            <dependency filterRef="8" />
        </dependencies>
        <filterDataInput location="dependency"
            dependencyFilterRef="8" />
    </digidatasignaturevalidator>

    <digidatasignatureremover id="10">
        <dependencies>
            <dependency filterRef="9" />
        </dependencies>
        <filterDataInput location="dependency"
            dependencyFilterRef="9" />
    </digidatasignatureremover>

    <custom id="11" type="CustomParser">
        <description>Custom MagParser filter.</description>
        <files>
            <file id="event" name="event" type="txt"
                file="domainPairs/sipr_nipr/dsg-filters2/11/
                    customParser.dat">
                <description>CustomParser for event data</description>
            </file>
        </files>
        <options>
            <option name="stripHeader">TRUE</option>
        </options>
        <dependencies>
            <dependency filterRef="10" />
        </dependencies>
        <filterDataInput location="dependency"
            dependencyFilterRef="10" />
    </custom>
</filters>
```

### 5.5.19 `<xmlvalidator>` element (subelement of `filters`)

This element is used to define a **Filter** to perform XML validation (sometimes incorrectly called schema validation). The required `id` attribute identifies the **Filter** within the

## UNCLASSIFIED

**Filterset.** (See schema for <filterset> for the schema xs:key definition.) Valid characters for id are alphanumeric, dashes, and underscores, and it must be 8 characters or less. The required schemaRef attribute references one of the **Schema** id's in the **Schemas** section.

This element contains an optional <description> element to provide a text description about the **Filter** to aid evaluators to better understand what it does.

See section 5.5.30 for description of optional element <options>.

See section 5.5.32 for description of optional element <dependencies>.

See section 5.5.34 for description of element <filterDataInput>.

schema:

```
<xs:element name="xmlvalidator">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="description" type="dfc:string"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="options" type="dfc:options" minOccurs="0"
        maxOccurs="1" />
      <xs:element name="dependencies" type="dfc:dependencies"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="filterDataInput"
        type="dfc:filterDataInput" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required" />
    <xs:attribute name="schemaRef" type="xs:string"
      use="required"/>
  </xs:complexType>

  <xs:key name="xmlvalidatorOptionsKey">
    <xs:selector xpath="dfc:options/dfc:option"></xs:selector>
    <xs:field xpath="@name" />
  </xs:key>
</xs:element>
```

example:

```
<xmlvalidator id="1" schemaRef="Event or Blog" >
  <filterDataInput location="original" />
</xmlvalidator>
```

### 5.5.20 <schematron> element (subelement of filters)

This element is used to define a **Filter** to perform a Schematron assertion. The required id attribute uniquely identifies the **Filter** within the **Filterset**. (See schema for <filterset> for the schema xs:key definition.) Valid characters for id are alphanumeric, dashes, and underscores, and it must be 8 characters or less. The required schematronRef attribute references one of the **Schematron** id's in the **Schematrons** section.

This element contains an optional <description> element to provide a text description

## UNCLASSIFIED

about the **Filter** to aid evaluators to better understand what it does.

See section 5.5.30 for description of optional element <options>.

See section 5.5.32 for description of optional element <dependencies>.

See section 5.5.34 for description of element <filterDataInput>.

schema:

```
<xs:element name="schematron">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="description" type="dfc:string"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="options" type="dfc:options" minOccurs="0"
        maxOccurs="1" />
      <xs:element name="dependencies" type="dfc:dependencies"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="filterDataInput"
        type="dfc:filterDataInput"
        minOccurs="1" maxOccurs="1" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required" />
    <xs:attribute name="schematronRef" type="xs:string"
      use="required" />
  </xs:complexType>

  <xs:key name="schematronOptionsKey">
    <xs:selector xpath="dfc:options/dfc:option" />
    <xs:field xpath="@name" />
  </xs:key>
</xs:element>
```

example:

```
<schematron id="7" schematronRef="example">
  <dependencies>
    <dependency filterRef="6" />
  </dependencies>
  <filterDataInput location="dependency"
    dependencyFilterRef="6" />
</schematron>
```

### 5.5.21 <xsltransformer> element (subelement of filters)

This element is used to define a **Filter** to perform a Stylesheet transformation. The required `id` attribute uniquely identifies the **Filter** within the **Filterset**. (See schema for <filterset> for the schema xs:key definition.) Valid characters for `id` are alphanumeric, dashes, and underscores, and it must be 8 characters or less. The required `stylesheetRef` attribute references one of the **Stylesheet** id's in the **Stylesheets** section.

This element contains an optional <description> element to provide a text description

## UNCLASSIFIED

about the **Filter** to aid evaluators to better understand what it does.

See section 5.5.30 for description of optional element <options>.

See section 5.5.32 for description of optional element <dependencies>.

See section 5.5.34 for description of element <filterDataInput>.

schema:

```
<xsl:element name="xsltransformer">
  <xsl:complexType>
    <xsl:sequence>
      <xsl:element name="description" type="dfc:string"
        minOccurs="0" maxOccurs="1" />
      <xsl:element name="options" type="dfc:options" minOccurs="0"
        maxOccurs="1" />
      <xsl:element name="dependencies" type="dfc:dependencies"
        minOccurs="0" maxOccurs="1" />
      <xsl:element name="filterDataInput"
        type="dfc:filterDataInput"
        minOccurs="1" maxOccurs="1" />
    </xsl:sequence>
    <xsl:attribute name="id" type="xs:string" use="required" />
    <xsl:attribute name="stylesheetRef" type="xs:string"
      use="required" />
  </xsl:complexType>
  <xsl:key name="xsltransformerOptionsKey">
    <xsl:selector xpath="dfc:options/dfc:option"/>
    <xsl:field xpath="@name" />
  </xsl:key>
</xsl:element>
```

example:

```
<xsltransformer id="6" stylesheetRef="Event Or Blog">
  <dependencies>
    <dependency filterRef="5" />
  </dependencies>
  <filterDataInput location="dependency"
    dependencyFilterRef="5" />
</xsltransformer>
```

### 5.5.22 <xslvalidator> element (subelement of filters)

This element is used to define a **Filter** to perform a Stylesheet validation. The required `id` attribute uniquely identifies the **Filter** within the **Filterset**. (See schema for <filterset> for the schema `xs:key` definition.) Valid characters for `id` are alphanumeric, dashes, and underscores, and it must be 8 characters or less. The required `stylesheetRef` attribute references one of the **Stylesheet** id's in the **Stylesheets** section.

This element contains an optional <description> element to provide a text description about the **Filter** to aid evaluators to better understand what it does.

## UNCLASSIFIED

See section 5.5.30 for description of optional element <options>.

See section 5.5.32 for description of optional element <dependencies>.

See section 5.5.34 for description of element <filterDataInput>.

schema:

```
<xsl:element name="xslvalidator">
  <xsl:complexType>
    <xsl:sequence>
      <xsl:element name="description" type="dfc:string"
        minOccurs="0" maxOccurs="1" />
      <xsl:element name="options" type="dfc:options" minOccurs="0"
        maxOccurs="1" />
      <xsl:element name="dependencies" type="dfc:dependencies"
        minOccurs="0" maxOccurs="1" />
      <xsl:element name="filterDataInput"
        type="dfc:filterDataInput"
        minOccurs="1" maxOccurs="1" />
    </xsl:sequence>
    <xsl:attribute name="id" type="xs:string" use="required" />
    <xsl:attribute name="stylesheetRef" type="xs:string"
      use="required" />
  </xsl:complexType>
  <xsl:key name="xslvalidatorOptionsKey">
    <xsl:selector xpath="dfc:options/dfc:option" />
    <xsl:field xpath="@name" />
  </xsl:key>
</xsl:element>
```

example:

```
<xslvalidator id="7" stylesheetRef="Event Or Blog">
  <dependencies>
    <dependency filterRef="6" />
  </dependencies>
  <filterDataInput location="dependency"
    dependencyFilterRef="6" />
</xslvalidator>
```

### 5.5.23 <wordchecker> element (subelement of filters)

This element is used to define a **Filter** to perform a dirty/clean word check. The required `id` attribute uniquely identifies the **Filter** within the **Filterset**. (See schema for <filterset> for the schema xs:key definition.) Valid characters for `id` are alphanumeric, dashes, and underscores, and it must be 8 characters or less. The required `file` attribute contains the name and full DFC relative path to a dirty/clean words XML file. (See section 5.6 for Dirty/Clean Words XML.)

This element contains an optional <description> element to provide a text description about the **Filter** to aid evaluators to better understand what it does.

## UNCLASSIFIED

See section 5.5.30 for description of optional element <options>.

See section 5.5.32 for description of optional element <dependencies>.

See section 5.5.34 for description of element <filterDataInput>.

schema:

```
<xs:element name="wordchecker">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="description" type="dfc:string"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="options" type="dfc:options" minOccurs="0"
        maxOccurs="1" />
      <xs:element name="dependencies" type="dfc:dependencies"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="filterDataInput"
        type="dfc:filterDataInput"
        minOccurs="1" maxOccurs="1" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required" />
    <xs:attribute name="file" type="xs:string" use="required" />
  </xs:complexType>
  <xs:key name="wordcheckerOptionsKey">
    <xs:selector xpath="dfc:options/dfc:option" />
    <xs:field xpath="@name" />
  </xs:key>
</xs:element>
```

example:

```
<wordchecker id="2" file="domainPairs/sipr_nipr/dsg-filters2/
  2/dirty-clean-words.xml">
  <dependencies>
    <dependency filterRef="1" />
  </dependencies>
  <filterDataInput location="dependency"
    dependencyFilterRef="1" />
</wordchecker>
```

### 5.5.24 <xmlnormalizer> element (subelement of filters)

This element is used to define a **Filter** to perform a XML normalization. The required `id` attribute uniquely identifies the **Filter** within the **Filterset**. (See schema for <filterset> for the schema xs:key definition.) Valid characters for `id` are alphanumeric, dashes, and underscores, and it must be 8 characters or less.

Normalization is defined as:

1. Tab, Carriage Returns, and Line Feeds are replaced with spaces. This only applies to XML element and attributes. Attribute and element content is not affected.
2. Whitespace that occurs consecutively more than once are removed. This only

## UNCLASSIFIED

applies to XML element and attributes. Attribute and element content is not affected.

3. All XML comments are removed

This element contains an optional <description> element to provide a text description about the **Filter** to aid evaluators to better understand what it does.

See section 5.5.30 for description of optional element <options>.

See section 5.5.32 for description of optional element <dependencies>.

See section 5.5.34 for description of element <filterDataInput>.

schema:

```
<xs:element name="xmlnormalizer">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="description" type="dfc:string"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="options" type="dfc:options" minOccurs="0"
        maxOccurs="1" />
      <xs:element name="dependencies" type="dfc:dependencies"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="filterDataInput"
        type="dfc:filterDataInput"
        minOccurs="1" maxOccurs="1" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required" />
  </xs:complexType>
  <xs:key name="xmlnormalizerOptionsKey">
    <xs:selector xpath="dfc:options/dfc:option" />
    <xs:field xpath="@name" />
  </xs:key>
</xs:element>
```

example:

```
<xmlnormalizer id="3">
  <dependencies>
    <dependency filterRef="2" />
  </dependencies>
  <filterDataInput location="dependency"
    dependencyFilterRef="2" />
</xmlnormalizer>
```

### 5.5.25 <classificationchecker> element (subelement of filters)

This element is used to define a **Filter** to perform a **Classification** check. It uses the **Classification** list defined in the **Domain** referenced in the toDomainRef of the current **Filterset**. The required **id** attribute uniquely identifies the **Filter** within the **Filterset**. (See schema for <filterset> for the schema xs:key definition.) Valid characters for **id** are alphanumeric, dashes, and underscores, and it must be 8 characters or less.

This element contains an optional <description> element to provide a text description about the **Filter** to aid evaluators to better understand what it does.

## UNCLASSIFIED

See section 5.5.30 for description of optional element <options>.

See section 5.5.32 for description of optional element <dependencies>.

See section 5.5.34 for description of element <filterDataInput>.

schema:

```
<xs:element name="classificationchecker">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="description" type="dfc:string"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="options" type="dfc:options" minOccurs="0"
        maxOccurs="1" />
      <xs:element name="dependencies" type="dfc:dependencies"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="filterDataInput"
        type="dfc:filterDataInput"
        minOccurs="1" maxOccurs="1" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required" />
  </xs:complexType>
  <xs:key name="classificationcheckerOptionsKey">
    <xs:selector xpath="dfc:options/dfc:option" />
    <xs:field xpath="@name" />
  </xs:key>
</xs:element>
```

example:

```
<classificationchecker id="4">
  <dependencies>
    <dependency filterRef="3" />
  </dependencies>
  <filterDataInput location="dependency"
    dependencyFilterRef="3" />
</classificationchecker>
```

### 5.5.26 <viruschecker> element (subelement of filters)

This element is used to define a **Filter** to perform a virus check. The required `id` attribute uniquely identifies the **Filter** within the **Filterset**. (See schema for <filterset> for the schema xs:key definition.) Valid characters for `id` are alphanumeric, dashes, and underscores, and it must be 8 characters or less.

This element contains an optional <description> element to provide a text description about the **Filter** to aid evaluators to better understand what it does.

See section 5.5.30 for description of optional element <options>.

See section 5.5.32 for description of optional element <dependencies>.

## UNCLASSIFIED

See section 5.5.34 for description of element <filterDataInput>.

schema:

```
<xs:element name="viruschecker">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="description" type="dfc:string"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="options" type="dfc:options" minOccurs="0"
        maxOccurs="1" />
      <xs:element name="dependencies" type="dfc:dependencies"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="filterDataInput"
        type="dfc:filterDataInput"
        minOccurs="1" maxOccurs="1" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required" />
  </xs:complexType>
  <xs:key name="viruscheckerOptionsKey">
    <xs:selector xpath="dfc:options/dfc:option" />
    <xs:field xpath="@name" />
  </xs:key>
</xs:element>
```

example:

```
<viruschecker id="5">
  <dependencies>
    <dependency filterRef="4" />
  </dependencies>
  <filterDataInput location="original"/>
</viruschecker>
```

### 5.5.27 <digitalSignaturevalidator> element (subelement of filters)

This element is used to define a **Filter** to perform a digital signature validation. The required `id` attribute uniquely identifies the **Filter** within the **Filterset**. (See schema for <filterset> for the schema xs:key definition.) Valid characters for `id` are alphanumeric, dashes, and underscores, and it must be 8 characters or less.

The required `source` attribute defines whether the digital signature is validated internally or externally. Valid values for `source` are: `external`, `internal`, and `both`. The optional <urls> element (required if `source` value is `external`) defines the external URLs (Uniform Resource Locator) of the validator. It contains one or more <url> elements, with required attribute `value` containing the URL.

Digital Signatures are assumed to comply to the World Wide Consortium XML Digital Signature specification.

This element contains an optional <description> element to provide a text description about the **Filter** to aid evaluators to better understand what it does.

See section 5.5.30 for description of optional element <options>.

See section 5.5.32 for description of optional element <dependencies>.

## UNCLASSIFIED

See section 5.5.34 for description of element <filterDataInput>.

schema:

```
<xs:element name="digitalSignatureValidator">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="urls" minOccurs="0" maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="description" type="dfc:string"
              minOccurs="0" maxOccurs="1" />
            <xs:element name="url" minOccurs="1"
              maxOccurs="unbounded">
              <xs:complexType>
                <xs:attribute name="value" type="xs:string"
                  use="required" />
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="options" type="dfc:options" minOccurs="0"
      maxOccurs="1" />
    <xs:element name="dependencies" type="dfc:dependencies"
      minOccurs="0" maxOccurs="1" />
    <xs:element name="filterDataInput"
      type="dfc:filterDataInput"
      minOccurs="1" maxOccurs="1" />
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required" />
  <xs:attribute name="source" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="internal" />
        <xs:enumeration value="external" />
        <xs:enumeration value="both" />
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  </xs:complexType>
  <xs:key name=" digitalSignatureValidatorOptionsKey">
    <xs:selector xpath="dfc:options/dfc:option" />
    <xs:field xpath="@name" />
  </xs:key>
</xs:element>
```

example:

```
<digitalSignatureValidator id="8" source="external">
  <urls>
    <url value="OCSP://myca.dod.mil" />
  </urls>

  <dependencies>
    <dependency filterRef="7" />
  </dependencies>
```

## UNCLASSIFIED

```
<filterDataInput location="dependency"
    dependencyFilterRef="7" />
</digitalSignatureValidator>
```

### 5.5.28 <*digitalSignatureRemover*> element (subelement of *filters*)

This element is used to define a **Filter** to perform a digital signature removal. The required `id` attribute uniquely identifies the **Filter** within the **Filterset**. (See schema for `<filterset>` for the schema `xs:key` definition.) Valid characters for `id` are alphanumeric, dashes, and underscores, and it must be 8 characters or less.

Digital Signatures are assumed to comply to the World Wide Consortium XML Digital Signature specification.

This element contains an optional `<description>` element to provide a text description about the **Filter** to aid evaluators to better understand what it does.

See section 5.5.30 for description of optional element `<options>`.

See section 5.5.32 for description of optional element `<dependencies>`.

See section 5.5.34 for description of element `<filterDataInput>`.

schema:

```
<xs:element name="digitalSignatureRemover">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="description" type="dfc:string"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="options" type="dfc:options" minOccurs="0"
        maxOccurs="1" />
      <xs:element name="dependencies" type="dfc:dependencies"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="filterDataInput"
        type="dfc:filterDataInput"
        minOccurs="1" maxOccurs="1" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required" />
  </xs:complexType>
  <xs:key name="digitalSignatureRemoverOptionsKey">
    <xs:selector xpath="dfc:options/dfc:option" />
    <xs:field xpath="@name" />
  </xs:key>
</xs:element>
```

example:

```
<digitalSignatureRemover id="9">
  <dependencies>
    <dependency filterRef="8" />
  </dependencies>
  <filterDataInput location="dependency"
    dependencyFilterRef="8" />
</digitalSignatureRemover>
```

### 5.5.29 <custom> element (subelement of filters)

The <custom> element provides a means for guard vendors to add additional types of **Filters** to **Filtersets** other than xmlvalidator, schematron, xsldtransformer, xsldvalidator, wordchecker, xmlnormalizer, classificationchecker and viruschecker. The required `id` attribute uniquely identifies the **Filter** within the **Filterset**. (See <filterset> for the schema xs:key definition.) Valid characters for `id` are alphanumeric, dashes, and underscores, and it must be 8 characters or less. The required `type` attribute defines the **Filter** type. There are no predefined values for type; instead, it is the guard's responsibility to read this attribute and understand how to map it to an internal filter, or potentially reject the DFC if there is no match.

This element contains an optional <description> element to provide a text description about the **Filter** to aid evaluators to better understand what it does.

This element contains optional <files> element to associate one or more files with the **Filter**. Each file is represented by a <file> element. The required `id` attribute must be unique within this **Filter**. The optional `name` attribute allows a more human-readable identifier to be displayed for the file. The optional `type` attribute provides a way for the guard to be able to know the file type. There are no predefined types; it is the responsibility of the guard to be able to interpret this value. The required `file` attribute contains the name and full DFC relative path to the actual file in the DFC ZIP. The element <file> also contains an optional <description> to provide a text description about the usage of the file for the **Filter**.

See section 5.5.30 for description of optional element <options>.

See section 5.5.32 for description of optional element <dependencies>.

See section 5.5.34 for description of element <filterDataInput>.

schema:

```
<xs:element name="custom">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="description" type="xs:string"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="files" minOccurs="0" maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="file" minOccurs="1"
              maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="description" type="xs:string"
                    minOccurs="0" maxOccurs="1"/>
                </xs:sequence>
                <xs:attribute name="id" use="required" />
                <xs:attribute name="name" use="optional" />
                <xs:attribute name="type" use="optional" />
                <xs:attribute name="file" use="required" />
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

## UNCLASSIFIED

```
</xs:sequence>
</xs:complexType>
<xs:key name="fileKey">
    <xs:selector xpath="dfc:file" />
    <xs:field xpath="@id" />
</xs:key>
</xs:element>
<xs:element name="options" type="dfc:options" minOccurs="0"
    maxOccurs="1" />
<xs:element name="dependencies" type="dfc:dependencies"
    minOccurs="0" maxOccurs="1" />
<xs:element name="filterDataInput"
    type="dfc:filterDataInput"
    minOccurs="1" maxOccurs="1" />
</xs:sequence>
<xs:attribute name="id" type="xs:string" use="required" />
<xs:attribute name="type" type="xs:string" use="required" />
</xs:complexType>
<xs:key name="customOptionsKey">
    <xs:selector xpath="dfc:options/dfc:option" />
    <xs:field xpath="@name" />
</xs:key>
</xs:element>
```

example:

```
<custom id="10" type="CustomParser">
    <description>Custom MagParser filter.</description>
    <files>
        <file id="event" name="event" type="txt"
            file="domainPairs/sipr_nipr/dsg-filters2/8/
            customParser.dat">
            <description>CustomParser for event data</description>
        </file>
    </files>
    <options>
        <option name="stripHeader">TRUE</option>
    </options>
    <dependencies>
        <dependency filterRef="9" />
    </dependencies>
    <filterDataInput location="dependency"
        dependencyFilterRef="9" />
</custom>
```

### 5.5.30 <options> element

All of the above **Filters** contain an optional <options> element to associate one or more name/value pairs with the **Filter**. This is typically used to provide minor configuration information to a filter. For example, you may have a custom filter in your filterset to handle a JPEG image and the image processing engine needs the upper and lower compression bounds of the JPEG compression engine.

schema:

```
<xs:complexType name="options">
```

## UNCLASSIFIED

```
<xs:sequence>
  [See 5.5.31 for definition of option element]
</xs:sequence>
</xs:complexType>
```

example:

```
<options>
  <option name="stripHeader">TRUE</option>
</options>
```

### 5.5.31 *<option> element (subelement of: options)*

This element contains one or more name/value pairs. The required `name` attribute must be unique within the scope of the **Filter**. The text element of `<option>` contains the value, allowing Unicode characters to be included in the value.

schema:

```
<xs:element name="option" minOccurs="1" maxOccurs="unbounded">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="name" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

example:

```
<option name="stripHeader">TRUE</option>
```

### 5.5.32 *<dependencies> element*

All of the above **Filters** contain an optional `<dependencies>` element that provides a way to support **Filter** ordering. It has a list of references to other **Filters** that are required to execute and succeed before this **Filter** may be executed.

schema:

```
<xs:element name="dependencies" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:sequence>
      [See 5.5.33 for definition of dependency element]
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

example:

```
<dependencies>
  <dependency filterRef="9" />
</dependencies>
```

## UNCLASSIFIED

### 5.5.33 <dependency> element (subelement of: dependencies)

This element contains one or more references to a **Filter** id. The required filterRef attribute must have a valid **Filter** id. It must not contain its own id, or the id of any other **Filter** that directly or indirectly depend on this **Filter**, causing circular dependencies.

schema:

```
<xs:element name="dependency" minOccurs="0"
  maxOccurs="unbounded">
  <xs:complexType>
    <xs:attribute name="filterRef" type="xs:string"
      use="required"/>
  </xs:complexType>
</xs:element>
```

example:

```
<dependency filterRef="9" />
```

### 5.5.34 <filterDataInput> element

All of the above **Filters** contain an optional <filterDataInput> element that describes the how a filter gets its input data. The required location attribute is an enumerated list and contains one of three values: dependency, original, and none. The dependency value indicates that the filter is to get its data input from a dependent filter. The dependencyFilterRef attribute indicates from which dependency the filter gets its input. The original value indicates that the filter is to get its data from the original input to the filterset (i.e. the data has not been processed by any other filters) and is usually used with filters that do not change data. The none value is used to indicate that the filter does not require any input and is used infrequently.

schema:

```
<xs:complexType name="filterDataInput">
  <xs:attribute name="location" use="required">
    <xs:simpleType>
      <xs:restriction base="xs:string">
        <xs:enumeration value="dependency"/>
        <xs:enumeration value="original"/>
        <xs:enumeration value="none"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:attribute>
  <xs:attribute name="dependencyFilterRef" type="xs:string"
    use="optional"/>
</xs:complexType>
</xs:element>
```

example:

```
<filterDataInput location="none" />
```

## UNCLASSIFIED

### 5.5.35 Domain Pairs Example

Cumulative example, including **Data Flow Metadata**, **Guards**, **Domains**, **Files**, and **Domain Pairs**:

```
<?xml version="1.0" encoding="UTF-8" ?>
<dfc
    xmlns:IC="urn:us:gov:ic:ism:v2"
    xmlns:ddms="http://metadata.dod.mil/mdr/ns/DDMS/1.3/"
    xmlns="http://cie.jfcom.mil/schema/dfc"
    version="1.2.11">

    <name>Biometrics</name>
    <uuid>75aa38db-9a0b-4cb4-8d0a-f1e95fc761e7</uuid>
    <version>1.0</version>
    <created>2008-01-01</created>
    <createdBy>snappc</createdBy>
    <modified>2008-01-04</modified>
    <modifiedBy>duncans</modifiedBy>
    <description>This is the DFC for Biometrics demo</description>
    <ddms:security IC:ownerProducer="USA" IC:classification="U" />
    <expirationDate>2010-01-01</expirationDate>
    <enableDate>2008-01-01</enableDate>
    <ticketRequest>2008-01-01</ticketRequest>

    <guard name="WSG" version="1.0" />
    <guard name="DSG" version="3.0">
        <xformConfig xforms="guardconfig/dsg-3.0.xhtml"
            css="guardconfig/dsg-3.0.css" instance="guardconfig/dsg-
            3.0-instance.xml" />
    </guard>

    <domains>
        <domain id="nipr">
            <description>This is the NIPR domain</description>
            <allowedInboundClassifications>
                <ddms:security
                    IC:ownerProducer="USA"
                    IC:classification="U" />
            </allowedInboundClassifications>
            <endpoints>
                <endpoint
                    id="NIPRWebService"
                    protocol="https"
                    ipAddress="100.101.102.103"
                    sendPort="8080"
                    type="server"
                    pathInfo="/service/WebService"
                    certificate="certificates/sipr-cert.pki">
                    <description>Web service on the NIPR
                        network</description>
                </endpoint>
            </endpoints>
        </domain>
        <domain id="sipr">
            <description>This is the SIPR domain</description>
            <minimumAllowedInboundClassifications>
```

## UNCLASSIFIED

```
<ddms:security IC:ownerProducer="USA"
    IC:classification="U" />
</minimumAllowedInboundClassifications>
<maximumAllowedInboundClassifications>
    <ddms:security IC:ownerProducer="USA"
        IC:classification="S" />
</maximumAllowedInboundClassifications>
<endpoints>
    <endpoint
        id="SIPRWebService"
        protocol="https"
        ipAddress="200.201.202.203"
        sendPort="8080"
        type="server"
        pathInfo="/service/WebService" />
</endpoints>
</domain>
</domains>

<files>
    <schemas>
        <schema id="Event Or Blog" type="W3C"
            file="schemas/Event Or Blog/addEventOrBlog.xsd">
            <description>This schema is for event or blog
                objects.</description>
        </schema>
        <schema id="Correlation Id" type="W3C"
            file="schemas/Correlation Id/correlationId.xsd" />
    </schemas>
    <schematrons>
        <schematron id="example" file="schematrons/
            example/example.strn">
            <description>This schematron is for event or blog
                objects.</description>
        </schematron>
        <schematron id="example2" file="schematrons/
            example2/example2.strn" />
    </schematrons>
    <stylesheets>
        <stylesheet id="Event Or Blog"
            file="stylesheets/Event Or Blog/addEventOrBlog.xsl">
            <description>This stylesheet is for event or blog
                objects.</description>
        </stylesheet>
        <stylesheet id="example" file="stylesheets/example/
            example.xsl" />
    </stylesheets>
    <others>
        <other id="Test Data" type="Test Data"
            file="others/Test Data/testData.xml" >
            <description>This is an example XML message to use as a
                test</description>
        </other>
    </others>
</files>

<domainPairs>
```

**UNCLASSIFIED**

```
<domainPair domain1Ref="sipr" domain2Ref="nipr">
  <filterset id="dsg-filters1" fromDomainRef="sipr"
    toDomainRef="nipr">
    <guard name="DSG" version="3.0">
      <xformConfig
        xforms="guardconfig/dsg-3.0-filterset.xhtml"
        css="guardconfig/dsg-3.0-filterset.css"
        instance="domainPairs/sipr_nipr/dsg-filters1
          /guardconfig/dsg-3.0-instance.xml" />
    </guard>

    <fromEndpoints>
      <endpoint ref="SIPRWebService" />
    </fromEndpoints>

    <toEndpoints>
      <endpoint ref="NIPRWebService" />
    </toEndpoints>

    <requires type="all">
      <require filtersetRef="dsg-filters2" />
    </requires>

    <associates>
      <associate filtersetRef="dsg-filters2" />
    </associates>

    <filters>
      <xmlvalidator id="1" schemaRef="Correlation Id">
        <filterDataInput location="original" />
      </xmlvalidator>
    </filters>
  </filterset>

  <filterset id="dsg-filters2" fromDomainRef="sipr"
    toDomainRef="nipr">
    <guard name="DSG" version="3.0">
      <options>
        <option name="max_filter_memory_size">1000000</option>
      </options>
      <xformConfig
        xforms="guardconfig/dsg-3.0-filterset.xhtml"
        css="guardconfig/dsg-3.0-filterset.css"
        instance="domainPairs/sipr_nipr/dsg-filters2
          /guardconfig/dsg-3.0-instance.xml" />
    </guard>

    <fromEndpoints>
      <endpoint ref="SIPRWebService" />
    </fromEndpoints>

    <toEndpoints>
      <endpoint ref="NIPRWebService" />
    </toEndpoints>

    <requires type="all">
      <require filtersetRef="dsg-filters1" />
    </requires>
  </filterset>
</domainPair>
```

**UNCLASSIFIED**

```
</requires>

<dependencies>
    <dependency filtersetRef="dsg-filters1" />
</dependencies>

<associates>
    <associate filtersetRef="dsg-filters1"/>
</associates>

<filters>
    <xmlvalidator id="1" schemaRef="Event or Blog" >
        <filterDataInput location="original" />
    </xmlvalidator>

    <wordchecker id="2" file="domainPairs/sipr_nipr/dsg-
        filters2/2/dirty-clean-words.xml">
        <dependencies>
            <dependency filterRef="1" />
        </dependencies>
        <filterDataInput location="dependency"
            dependencyFilterRef="1" />
    </wordchecker>

    <xmlnormalizer id="3">
        <dependencies>
            <dependency filterRef="2" />
        </dependencies>
        <filterDataInput location="dependency"
            dependencyFilterRef="2" />
    </xmlnormalizer>

    <classificationchecker id="4">
        <dependencies>
            <dependency filterRef="3" />
        </dependencies>
        <filterDataInput location="dependency"
            dependencyFilterRef="3" />
    </classificationchecker>

    <viruschecker id="5">
        <dependencies>
            <dependency filterRef="4" />
        </dependencies>
        <filterDataInput location="original"/>
    </viruschecker>

    <xsltransformer id="6" stylesheetRef="Event Or Blog">
        <dependencies>
            <dependency filterRef="5" />
        </dependencies>
        <filterDataInput location="dependency"
            dependencyFilterRef="5" />
    </xsltransformer>

    <xslvalidator id="7" stylesheetRef="Event Or Blog">
        <dependencies>
```

**UNCLASSIFIED**

```
<dependency filterRef="6" />
</dependencies>
<filterDataInput location="dependency"
    dependencyFilterRef="6" />
</xslvalidator >

<schematron id="8" schematronRef="example">
    <dependencies>
        <dependency filterRef="7" />
    </dependencies>
    <filterDataInput location="dependency"
        dependencyFilterRef="7" />
</schematron>

<digidatavalidator id="9" source="external">
    <urls>
        <url value="OCSP://myca.dod.mil"/>
    </urls>

    <dependencies>
        <dependency filterRef="8" />
    </dependencies>
    <filterDataInput location="dependency"
        dependencyFilterRef="8" />
</digidatavalidator>

<digidataremover id="10">
    <dependencies>
        <dependency filterRef="9" />
    </dependencies>
    <filterDataInput location="dependency"
        dependencyFilterRef="9" />
</digidataremover>

<custom id="11" type="CustomParser">
    <description>Custom MagParser filter.</description>
    <files>
        <file id="event" name="event" type="txt"
            file="domainPairs/sipr_nipr/dsg-filters2/11/
            customParser.dat">
            <description>CustomParser for event
                data</description>
        </file>
    </files>
    <options>
        <option name="stripHeader">TRUE</option>
    </options>
    <dependencies>
        <dependency filterRef="10" />
    </dependencies>
    <filterDataInput location="dependency"
        dependencyFilterRef="10" />
    </custom>
</filters>
</filterset>
</domainPair>
</domainPairs>
```

## UNCLASSIFIED

```
</dfc>
```

### 5.6 Dirty/Clean Words XML

The Dirty/Clean Words XML is a file containing Dirty/Clean word lists for multiple languages. These files are used by the **Domain Pair Filterset's** `<wordchecker>` element. The file names are defined by the `<wordchecker>` elements' file attribute. The root element is `<dirtyCleanWordLists>` and contains multiple `<dirtyCleanWordList>` elements.

schema:

```
<?xml version="1.0"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  <xs:element name="dirtyCleanWordLists">
    <xs:complexType>
      <xs:sequence>
        [See 5.6.1 for definition of description element]
        [See 5.6.2 for definition of dirtyCleanWordsList element]
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

example:

```
<?xml version="1.0" encoding="UTF-8" ?>
<dirtyCleanWordLists
  xmlns="http://cie.jfcom.mil/schema/dfc">
  <description>Dirty/Clean Words to be used with the header in
  SIPR,NIPR dsg-filters</description>
  <dirtyCleanWordList lang="en">
    <dirtyWords file="domainPairs/sipr_nipr/dsg-filters2/2/
      dirtyWords.txt"/>
    <cleanWords file="domainPairs/sipr_nipr/dsg-filters2/2/
      cleanWords.txt"/>
    <ignoreCharacters file="domainPairs/sipr_nipr/dsg-
      filters2/2/ignoreCharacters.txt"/>
  </dirtyCleanWordList>
  <dirtyCleanWordList lang="es">
    <dirtyWords file="domainPairs/sipr_nipr/dsg-filters2/2/es-
      dirtyWords.txt"/>
    <cleanWords file="domainPairs/sipr_nipr/dsg-filters2/2/es-
      cleanWords.txt"/>
    <ignoreCharacters file="domainPairs/sipr_nipr/dsg-
      filters2/2/es-ignoreCharacters.txt"/>
  </dirtyCleanWordList>
</dirtyCleanWordLists>
```

#### 5.6.1 `<description>` element

This element contains the text description of this Dirty/Clean Words XML file to allow the

## UNCLASSIFIED

end user to identify the Dirty/Clean Word List when it is being manually updated on the guard.

schema:

```
<xs:element name="description" type="xs:string" />
```

example:

```
<description>Dirty/Clean Words to be used with the header in  
SIPR,NIPR dsg-filters</description>
```

### 5.6.2 <dirtyCleanWordList> element

This element contain one or more **Dirty/Clean Words** for a specific language. The required lang attribute specifies the two-character International Standards Organization (ISO) language code for the dirty/clean words. Each Dirty/Clean Words List has dirty words, optional clean words, and an optional ignore character list.

schema:

```
<xs:element name="dirtyCleanWordList" minOccurs="0"  
maxOccurs="unbounded">  
<xs:complexType>  
<xs:sequence>  
    [See 5.6.3 for definition of dirtyWords element]  
    [See 5.6.4 for definition of cleanWords element]  
    [See 5.6.5 for definition of ignoreCharacters element]  
</xs:sequence>  
<xs:attribute name="lang" type="xs:string" use="required"/>  
</xs:complexType>  
</xs:element>
```

example:

```
<dirtyCleanWordList lang="en">  
    <dirtyWords file="domainPairs/sipr_nipr/dsg-  
        filters2/2/dirtyWords.txt"/>  
    <cleanWords file="domainPairs/sipr_nipr/dsg-  
        filters2/2/cleanWords.txt"/>  
    <ignoreCharacters file="domainPairs/sipr_nipr/dsg-  
        filters2/2/ignoreCharacters.txt"/>  
</dirtyCleanWordList>  
<dirtyCleanWordList lang="es">  
    <dirtyWords file="domainPairs/sipr_nipr/dsg-filters2/2/es-  
        dirtyWords.txt"/>  
    <cleanWords file="domainPairs/sipr_nipr/dsg-filters2/2/es-  
        cleanWords.txt"/>  
    <ignoreCharacters file="domainPairs/sipr_nipr/dsg-  
        filters2/2/es-ignoreCharacters.txt"/>  
</dirtyCleanWordList>
```

### 5.6.3 <dirtyWords> element (subelement of: dirtyCleanWordList)

## UNCLASSIFIED

This element contains the filename for the required dirty words list. The required `file` attribute contains the full DFC relative path and name of a text file containing a list of dirty words. The dirty word list file is in UTF-8 format. Each word is on its own line in the file. For English, the default filename is `dirtyWords.txt`, and for every other language the filename is `dirtyWords.txt` with the two-character language code and a dash pre-pended to it (for example, Korean would be `ko-dirtyWords.txt`).

schema:

```
<xs:element name="dirtyWords" minOccurs="1" maxOccurs="1">
  <xs:complexType>
    <xs:attribute name="file" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
```

example:

```
<dirtyWords file="domainPairs/sipr_nipr/dsg-filters/2/
  dirtyWords.txt"/>
```

### 5.6.4 `<cleanWords>` element (subelement of: `dirtyCleanWordList`)

This element contains the filename for the optional clean words list. Clean words are words that completely contain a dirty word. For example, the dirty word “secret” is contained in “secretary” so “secretary” would be considered a clean word. If clean words are not used, then it is highly likely that a number of false positives for dirty words will occur on the guard. The required `file` attribute contains the full DFC relative path and name of a text file containing a list of clean words. The clean word list file is in UTF-8 format. Each word is on its own line in the file. For English, the default filename is `cleanWords.txt`, and for every other language the filename is `cleanWords.txt` with the two-character language code and a dash pre-pended to it (for example, Korean would be `ko-cleanWords.txt`)

schema:

```
<xs:element name="cleanWords" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:attribute name="file" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
```

example:

```
<cleanWords file="domainPairs/sipr_nipr/dsg-filters/2/
  cleanWords.txt"/>
```

### 5.6.5 `<ignoreCharacters>` element (subelement of: `dirtyCleanWordList`)

## UNCLASSIFIED

This element contains the filename for the optional ignore characters. The required `file` attribute contains the full DFC relative path and name of a text file containing a list of characters to be ignored in the dirty and clean words lists. The ignore characters file is in UTF-8 format. Each ignore character is on its own line in the file. For English, the default filename is `ignoreCharacters.txt`, and for every other language the filename is `ignoreCharacters.txt` with the two-character language code and a dash pre-pended to it (for example, Korean would be `ko-ignoreCharacters.txt`)

schema:

```
<xs:element name="ignoreCharacters" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:attribute name="file" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
```

example:

```
<ignoreCharacters file="domainPairs/sipr_nipr/dsg-filters/2/
  ignoreCharacters.txt"/>
```

## 6. Digital Signing

The Digital Signing capability in the DFCF provides two functions: non-repudiation and tamper detection during the transmission and handling of the DFCF and, if implemented on the guard, the ability to restrict loading of DFCFs that have not been digitally signed by an authorized approver/certifier. Guards can also use this capability to determine if the file has been modified in transit by checking the digitally signed hash values of the files within the DFCF. The digital signatures are stored in the `digitalSignings.xml` and it is located at the root level of the DFC. The root element is `<digitalSigning>`. The element contains multiple file entries with their hash, a signing date, signed by, a signing status and a digital signature.

schema:

```

<?xml version="1.0"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  targetNamespace="http://cie.jfcom.mil/schema/dfc"
  xmlns:dfc="http://cie.jfcom.mil/schema/dfc"
  elementFormDefault="qualified">
  <xs:import
    namespace="http://www.w3.org/2000/09/xmldsig#"
    schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-
      schema.xsd" />
  <xs:element name="digitalSigning">
    <xs:complexType>
      <xs:sequence>
        [ See 6.1 for definition of fileEntry element]
        [ See 6.2 for definition of signed element]
        [ See 6.3 for definition of signedBy element]
        [ See 6.4 for definition of signStatus element]
        [ See 6.5 for definition of ds:Signature element]
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

example:

```

<?xml version="1.0" encoding="UTF-8"?>
<digitalSigning xmlns="http://cie.jfcom.mil/schema/dfc"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://cie.jfcom.mil/schema/dfc digitalSignatures.xsd"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

  <fileEntry filename="bray.xml"
    hash="07126db83b6a4df663be70d47eb30e74865fc26894569ba7792be078dfcbb04d"/>
  <fileEntry filename="dfc.xml"
    hash="9d8a3b5f49579e6217a85a86c56b16cd890cc2ceda2a6f9dc61ee99f27cd6b9a"/>

  <signed>2008-02-12</signed>
  <signedBy>CN=cg.reldomain.com,OU=SED,O=Trident Systems Inc,L=Morrisville,
    ST=North Carolina,C=US</signedBy>
  <signerRole>developer</signerRole>

  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>

```

## UNCLASSIFIED

```
<ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"></ds:CanonicalizationMethod>
<ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"></ds:SignatureMethod>
<ds:Reference URI="">
    <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"></ds:Transform>
        <ds:Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"></ds:Transform>
    </ds:Transforms>
    <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"></ds:DigestMethod>
        <ds:DigestValue>2YES0AiNrIFjYgb9f2qoa4btXAk=</ds:DigestValue>
    </ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>
    MIIJ... (large base64 encoded string)
</ds:SignatureValue>
<ds:KeyInfo>
    <ds:X509Data>
        <ds:X509Certificate>
            pASBsfi6P5ghqkbg1QT6q+c4bppUgDaHQvvMBFPsvw== (base64 encoded certificate)
        </ds:X509Certificate>
    </ds:X509Data>
<ds:KeyName>10:CF:C6:26:DA:04:12:58:B1:6D:A8:88:99:B0:DB:B6</ds:KeyName>
</ds:KeyInfo>
</ds:Signature>
</digitalSigning>
```

### 6.1    <fileEntry> element

This element contains one or more **File entries** to be signed in the DFC. The required `filename` contains the full DFC path and name of file entry. The required `hash` attribute contains a generated hash code for the file entry. The required hashing function is SHA-256.

## UNCLASSIFIED

schema:

```
<xs:element name="fileEntry" minOccurs="1" maxOccurs="unbounded">
  <xs:complexType>
    <xs:attribute name="filename" type="xs:string" use="required"/>
    <xs:attribute name="hash" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
```

example:

```
<fileEntry filename="bray.xml"
  hash="07126db83b6a4df663be70d47eb30e74865fc26894569ba7792be078dfcbb04d" />
<fileEntry filename="dfc.xml"
  hash="9d8a3b5f49579e6217a85a86c56b16cd890cc2ceda2a6f9dc61ee99f27cd6b9a" />
```

### 6.2 *<signed> element*

This element contains the date when the DFC ZIP was signed (format: CCYY-MM-DD)

schema:

```
<xs:element name="signed" type="xs:date" minOccurs="1" maxOccurs="1" />
```

example:

```
<signed>2008-02-12</signed>
```

### 6.3 *<signedBy> element*

This element contains the name of the signer of the DFCF. Its value is taken from the signature file and is usually the Distinguished Name (DN) of the user. It uses the format defined in RFC 2253. (See <http://www.ietf.org/rfc/rfc2253.txt?number=2253> )

schema:

```
<xs:element name="signedBy" type="xs:string" minOccurs="1" maxOccurs="1" />
```

example:

```
<signedBy>CN=cg.reldomain.com,OU=SED,O=Trident Systems Inc,L=Morrisville,
ST=North Carolina,C=US</signedBy>
```

### 6.4 *<signerRole> element*

This element contains the role of the signer of the DFCF. Valid values are: developer, submitter, evaluator, and approver.

schema:

```
<xs:element name="signerRole" minOccurs="1" maxOccurs="1">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="developer"/>
      <xs:enumeration value="submitter"/>
      <xs:enumeration value="evaluator"/>
      <xs:enumeration value="approver" />
    </xs:restriction>
```

## UNCLASSIFIED

```
</xs:simpleType>  
</xs:element>
```

example:

```
<signerRole>developer</signerRole>
```

### 6.5 <ds:Signature> element

This element contains the digital signature of the person signing the DFCF. There must be only one <ds:Signature> node containing a W3.org digital signature. See <http://www.w3.org/Signature> for details on digital signatures in XML.

schema:

```
<xs:element ref="ds:Signature"/>
```

example:

```
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">  
  <ds:SignedInfo>  
    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/  
      2001/REC-xml-c14n-20010315"></ds:CanonicalizationMethod>  
    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/  
      xmldsig#rsa-sha1"></ds:SignatureMethod>  
    <ds:Reference URI="">  
      <ds:Transforms>  
        <ds:Transform Algorithm="http://www.w3.org/2000/09/  
          xmldsig#enveloped-signature"></ds:Transform>  
        <ds:Transform Algorithm="http://www.w3.org/TR/2001/  
          REC-xml-c14n-20010315"></ds:Transform>  
      </ds:Transforms>  
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/  
        xmldsig#sha1"></ds:DigestMethod>  
      <ds:DigestValue>2YES0AiNrIFjYgb9f2qoa4btXAk=</ds:DigestValue>  
    </ds:Reference>  
  </ds:SignedInfo>  
  <ds:SignatureValue>  
    <ds:KeyInfo>  
      <ds:X509Data>  
        <ds:X509Certificate>  
MIIDc jCCAtugAwIBAgIJAlytAxOp1i/rMA0GCSqGSIb3DQEBBQUAMIGDMQswCQYDVQQGEwJVUzEX  
MBUGA1UECBMOTm9ydgGgQ2Fyb2xpbmExFDASBgnVBAcTC01vcnJpc3ZpbGx1MRwwGgYDVQQKExNU  
cmklZW50IFN5c3R1bxMgSW5jM0wwCgYDVQQLEwNTRUQxGTAXBgNVBAMTEGNnLnJ1bGRvbWFpbij  
b20wHhcNMDCwMzA3MjEwMzMwWhcNMDgwMzA2MjEwMzMwWjCBgzELMAkGA1UEBhMCVVMxFzAVBqNV  
BAgTDk5vcnRoIENhcm9saW5hMRQwEgYDVQQHEwtNb3JyaXN2aWxsZTEcMB0GA1UEChMTVHJpZGVu  
dCBTeXN0ZW1zIEluYzEMMAoGA1UECxMDU0VEMRkwFwYDVQQDEXBjZy5yZWxkb21haW4uY29tMIGf  
MA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDoyKT5/Vvg2uDXMSSv3M4jp20Vq0bqKHiP6of336m7  
mdwnl3xbqo55ufYUs2Iqp0V0O+6bg0p2X8+GhNfUKexQUTEb0Z1ktWjQDXmI4KQy/BPUmNytgcZH  
5wheelOUivNPiQQdhG1eUwFcwsAYweM/6dDnfN9Ui8NTrCCAYAbHAQIDAQABo4HrMIHoMB0GA1Ud  
DgQWBStTETHYoeoNUZVdaTRDongUcn06jCBuAYDVR0jBIGwMIGtgBStTETHYoeoNUZVdaTRDong  
Ucn06qGBiasBhjCBgzELMAkGA1UEBhMCVVMxFzAVBqNVBAgTDk5vcnRoIENhcm9saW5hMRQwEgYD  
VQQHEwtNb3JyaXN2aWxsZTEcMB0GA1UEChMTVHJpZGVudCBTeXN0ZW1zIEluYzEMMAoGA1UECxMD  
U0VEMRkwFwYDVQQDEXBjZy5yZWxkb21haW4uY29tggkAti0DE6nWL+swDAYDVR0TBAUwAwEB/zAN
```

**UNCLASSIFIED**

```
BgkqhkiG9w0BAQUFAAOBgQBmONxa4V9a94ZI9Ma1DcMj6oQRb5NGe/yIqPluYxwQs9s++HgQX+6Q  
z92tMNXNGiq5CEWVFN9j5xdQJynwlUQ8yVEvYBshwyxQq1Swbcyxn1AckySk6797EjHGmEbzlQmUpASBsfi6P5ghqkbg1QT6q+c4bppUgDaHQvvMBFPsvw==  
    </ds:X509Certificate>  
    </ds:X509Data>  
<ds:KeyName>10:CF:C6:26:DA:04:12:58:B1:6D:A8:88:99:B0:DB:B6</ds:KeyName>  
    </ds:KeyInfo>  
</ds:Signature>
```

## UNCLASSIFIED

### 7. Comments XML

Comments.xml contains a time-ordered list of comments, which may be individually digitally signed. Comments provide a simple mechanism to communicate and track workflow-type information between a developer, submitter, evaluator, and approver of an XML Data Flow. Some possible uses include the following: it can be used to explain why a schema rating is high and how it is being mitigated by a filter or another function in a guard; it can be used by an evaluator in seeking clarification from the developer on why the schema has a particular set of elements; or it can be an approver to state stipulations on how the Data Flow can be used. The root element is <comments>.

schema:

```
<?xml version="1.0"?>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
    targetNamespace="http://cie.jfcom.mil/schema/dfc"
    xmlns:dfc="http://cie.jfcom.mil/schema/dfc"
    elementFormDefault="qualified">

    <xs:import
        namespace="http://www.w3.org/2000/09/xmldsig#"
        schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-
        schema.xsd"/>

    <xs:element name="comments">
        <xs:complexType>
            <xs:sequence>
                [ See 7.1 for definition of comment element]
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

example:

```
<?xml version="1.0" encoding="UTF-8"?>
<comments xmlns="http://cie.jfcom.mil/schema/dfc">
    <comment author="schillingj" time="2008-02-15T09:39:00">
        <text>Boyd, can you give an example of what a comment should look
        like?</text>
    </comment>

    <comment author="fletcherb" time="2008-02-15T10:00:00">
        <text>Just write whatever you want into the comments text.</text>
        <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
            <ds:SignedInfo>
                <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/
                    2001/REC-xml-c14n-20010315"></ds:CanonicalizationMethod>
                <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/
                    xmldsig#rsa-sha1"></ds:SignatureMethod>
                <ds:Reference URI="">
                    <ds:Transforms>
                        <ds:Transform Algorithm="http://www.w3.org/2000/09/
                            xmldsig#enveloped-signature"></ds:Transform>
                        <ds:Transform Algorithm="http://www.w3.org/TR/2001/
                            REC-xml-c14n-20010315"></ds:Transform>
                    </ds:Transforms>
                </ds:Reference>
            </ds:SignedInfo>
        </ds:Signature>
    </comment>
</comments>
```

## UNCLASSIFIED

```
</ds:Transforms>
<ds:DigestMethod Algorithm="http://www.w3.org/2000/09/
    xmldsig#sha1"></ds:DigestMethod>
<ds:DigestValue>2YES0AiNrIFjYgb9f2qoa4btXAk=</ds:DigestValue>
</ds:Reference>
</ds:SignedInfo>
<ds:SignatureValue>
4pnNCxbmk+C9Hd3gMATbBAG+/CoSk+558s3UuZR4Jq95En4pxgnK5mpDvAFoPHDxTPh/U4MC5AJ5
3UXY78byIk0olu6spNc16jKzYSrp5cv8eQuXNz/PCggUYTGP0BSgUuunnfIEHzUe60Bg+TFO4hey
Fn18zsj7ZLhxXiMkSj8=
</ds:SignatureValue>
<ds:KeyInfo>
<ds:X509Data>
<ds:X509Certificate>
MIIDcjCCAtugAwIBAgIJALYtAxOp1i/rMA0GCSqGSIb3DQEBCQUAMIGDMQswCQYDVQQGEwJVUzEX
MBUGA1UECBMOTm9ydGggQ2Fyb2xpbmExFDASBgnNVBAcTC01vcnJpc3ZpbGx1MRwwGgYDVQQKExNU
cmlkZW50IFN5c3R1bXNgSW5jMQwwCgYDVQQLEwNTRUQxGTAXBgNVBAMTEGNnLnJ1bGRvbWFpbij
b20wHhcNMDCwMzA3MjEwMzMwWhcNMdgwMzA2MjEwMzMwWjCBgzELMAkGA1UEBhMCVVMxFzAVBgNV
BAgTDk5vcnRoIENhcm9saW5hMRQwEgYDVQQHEwtNb3JyaXN2aWxsZTEcMBoGA1UEChMTVHJpZGVu
dCBTeXN0ZW1zIEluYzEMMAoGA1UECxMDU0VERMkwFwYDVQQDEXBjZy5yZWxkb21haW4uY29tMIGf
MA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBqODoyKT5/Vvg2uDXMSSV3M4jP20Vq0bqKHiP6of336m7
mdwnl3xbqo55ufYUs2Iqp0V00+6bg0p2X8+GhNfUkexQUTEb0Z1KtWjQDXmI4KQy/BPUmNytcgZH
5whe1tOUivNPiQQdhG1eUwFcwsAYweM/6dDnfN9Ui8NTrCCAYAbHAQIDAQABo4HrMIHoMB0GA1UD
DgQWBStTETHYoeoNUZVdaTRDongUcn06jCBuAYDVR0jBIGwMIGtgBStTETHYoeoNUZVdaTRDong
Ucn06qGBiasBhjCBgzELMAkGA1UEBhMCVVMxFzAVBgNVBAgTDk5vcnRoIENhcm9saW5hMRQwEgYD
VQQHEwtNb3JyaXN2aWxsZTEcMBoGA1UEChMTVHJpZGVudCBTeXN0ZW1zIEluYzEMMAoGA1UECxMD
U0VERMkwFwYDVQQDEXBjZy5yZWxkb21haW4uY29tggkAti0DE6nWL+swDAYDVR0TBauAwEB/zAN
BgkqhkiG9w0BAQUFAOBgQBmONxa4V9a94ZI9Ma1DcMj6oQRb5NGe/yIqPluYxwQs9s++HgQX+6Q
z92tMNXNGiq5CEWFN9j5xdQJynw1UQ8yVEvYBshwyxQqlSwbcyxn1AckySk6797EjhGmEbziQmU
pASBsfi6P5ghqkgb1QT6q+c4bppUgDaHQvvMBFPsvw==
</ds:X509Certificate>
</ds:X509Data>
<ds:KeyName>10:CF:C6:26:DA:04:12:58:B1:6D:A8:88:99:B0:DB:B6</ds:KeyName>
</ds:KeyInfo>
</ds:Signature>
</comment>

<comment author="schillingj" time="2008-02-15T10:30:00">
    <text>Ok, thanks.</text>
</comment>
</comments>
```

### 7.1    *<comment> element (subelement of: comments)*

This element contains one or more comments. The required `author` attribute defines the name of the user entering the comment. The required `time` attribute has the date and time that the change occurred (format: CCYY-MM-DDTHH:MM:SS)

It has a required `<text>` element containing the body of the comment text. It has an optional `<ds:Signature>` element to allow a digital signature to be attached to the comment.

## UNCLASSIFIED

schema:

```
<xs:element name="comment" minOccurs="1" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="text" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element ref="ds:Signature" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
    <xs:attribute name="author" type="xs:string" use="required" />
    <xs:attribute name="time" type="xs:dateTime" use="required" />
  </xs:complexType>
</xs:element>
```

example:

```
<comment author="schillingj" time="2008-02-15T11:44:00">
  <text>This is an example comment.</text>
</comment>
```

## UNCLASSIFIED

### 8. Change Log

The file "changeLog.xml" contains the Change Log. Each change made to any of the files in the DFC ZIP is recorded as a message in the Change Log. It is located at the root level in the DFC ZIP. The root element is <changeLog>.

schema:

```
<?xml version="1.0"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://cie.jfcom.mil/schema/dfc"
  xmlns:dfc="http://cie.jfcom.mil/schema/dfc"
  elementFormDefault="qualified">

  <xs:element name="changeLog">
    <xs:complexType>
      <xs:sequence>
        [ See 8.1 for definition of message element]
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

example:

```
<?xml version="1.0" encoding="UTF-8"?>
<changeLog
  xmlns="http://cie.jfcom.mil/schema/dfc">

  <message author="schillingj" time="2008-02-13T10:57:00">
    Created new DFCF.</message>
  <message author="schillingj" time="2008-02-13T10:57:00">
    Set author to "schillingj"</message>
  <message author="schillingj" time="2008-02-13T12:00:00">
    Modified Name to "Biometrics"</message>
  <message author="schillingj" time="2008-02-13T12:05:00">
    Set classification level to "UNCLASSIFIED"</message>
  <message author="schillingj" time="2008-02-13T12:10:00">
    Modified Description to "Fake biometric service".</message>
  <message author="schillingj" time="2008-02-13T15:00:00">
    Set Guard to "WSG 1.0"</message>
  <message author="schillingj" time="2008-02-13T17:00:00">
    Added classification "UNCLASSIFIED"</message>
  <message author="schillingj" time="2008-02-13T17:05:00">
    Added classification "SECRET//X1"</message>
  <message author="schillingj" time="2008-02-13T17:10:00">
    Added classification "TOP SECRET///REL TO US AUS"</message>
  <message author="schillingj" time="2008-02-13T17:15:00">
    Added domain "SIPR"</message>
  <message author="schillingj" time="2008-02-13T17:20:00">
    Added domain "NIPR"</message>
  <message author="schillingj" time="2008-02-13T17:25:00">
    Assigned classification "UNCLASSIFIED" to SIPR</message>
</changeLog>
```

## UNCLASSIFIED

### 8.1 <message> element

This element contains a text message describing a single change. The required `author` attribute defines the user who made the change. The required `time` attribute has the date and time that the change occurred (format: CCYY-MM-DDTHH:MM:SS)

schema:

```
<xs:element name="message" minOccurs="1" maxOccurs="unbounded">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="author" type="xs:string"
          use="required" />
        <xs:attribute name="time" type="xs:dateTime"
          use="required" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

example:

```
<message author="schillingj" time="2008-02-13T17:25:00">
  Assigned classification "UNCLASSIFIED" to SIPR</message>
```

## 9. XForms (Guard Customization Information)

Guard Customization Information for the **Data Flow** and for **Filtersets** is accomplished by XForms. Bray supports the standard XHTML tags, with special notations to enable DFC support. To provide guard-specific properties, the guard vendor must create the XForms form definition file and an instance file. The instance file contains the structure of the desired XML properties and any default values. The vendor can also optionally provide a CSS to format the display of the XForms form in Bray.

### 9.1 DFC Model

All tags in the **Data Flow** can be accessed using the notation \${dfcModel}. At runtime this notation is replaced with the machine-appropriate URL to access the **Data Flow** tags.

example:

```
<xforms:model id="dfcf">
  <xforms:instance id="dfcf" src="${dfcModel}" xmlns="" />
</xforms:model>
```

### 9.2 Guard-Specific Properties

Guard-Specific Properties default values (or previously saved values) are imported into the XForms form using the notation \${instance}. At runtime this notation is replaced with the machine-appropriate URL to access the initial or saved instance. The notation \${submit} is replaced at runtime with the machine-appropriate URL to submit the current XForms instance and save it back into the DFC.

example:

```
<xforms:model id="model">
  <xforms:instance id="instance" src="${instance}" xmlns="" />
  <xforms:submission id="s" method="put" action="${submit}" />
</xforms:model>
```

**APPENDIX A - SCHEMAS****DFC.xsd**

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:ddms="http://metadata.dod.mil/ldr/ns/DDMS/1.3/"
  xmlns:dfc="http://cie.jfc.com.mil/schema/dfc"
  targetNamespace="http://cie.jfc.com.mil/schema/dfc"
  elementFormDefault="qualified">
  <xs:import namespace="http://metadata.dod.mil/ldr/ns/DDMS/1.3/"
    schemaLocation="http://metadata.dod.mil/ldr/ns/DDMS/1.3/" />
  <xs:element name="dfc">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name" type="xs:string" minOccurs="1"
          maxOccurs="1"/>
        <xs:element name="uuid" type="xs:string" minOccurs="1"
          maxOccurs="1"/>
        <xs:element name="version" type="xs:string" minOccurs="1"
          maxOccurs="1"/>
        <xs:element name="created" type="xs:date" minOccurs="1"
          maxOccurs="1"/>
        <xs:element name="createdBy" type="xs:string" minOccurs="1"
          maxOccurs="1"/>
        <xs:element name="modified" type="xs:date" minOccurs="1"
          maxOccurs="1"/>
        <xs:element name="modifiedBy" type="xs:string" minOccurs="1"
          maxOccurs="1"/>
        <xs:element name="description" type="xs:string" minOccurs="0"
          maxOccurs="1"/>
        <xs:element ref="ddms:security" minOccurs="1" maxOccurs="1"/>
        <xs:element name="expirationDate" type="xs:date" minOccurs="0"
          maxOccurs="1"/>
        <xs:element name="enableDate" type="xs:date" minOccurs="0"
          maxOccurs="1"/>
        <xs:element name="ticketRequest" type="xs:string" minOccurs="0"
          maxOccurs="1"/>

        <xs:element name="guard" minOccurs="0" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="options" type="dfc:options"
                minOccurs="0" maxOccurs="1" />
              <xs:element name="xformConfig" minOccurs="0" maxOccurs="1">
                <xs:complexType>
                  <xs:attribute name="xforms" type="xs:string" use="required"/>
                  <xs:attribute name="css" type="xs:string" use="optional"/>
                  <xs:attribute name="instance" type="xs:string"
                    use="required"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

## UNCLASSIFIED

```
<xs:attribute name="name" type="xs:string" use="required"/>
<xs:attribute name="version" type="xs:string" use="required"/>
<xs:attribute name="nativeSupport" type="xs:boolean"
    use="optional"/>
</xs:complexType>
<xs:key name="guardOptionsKey">
    <xs:selector xpath="dfc:options/dfc:option" />
    <xs:field xpath="@name" />
</xs:key>
</xs:element>

<xs:element name="domains" minOccurs="0" maxOccurs="1">
    <xs:complexType>
        <xs:sequence>
            <xs:element name ="domain" minOccurs="2" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="description" minOccurs="0" maxOccurs="1"/>
                        <xs:element name="allowedInboundClassifications"
                            minOccurs="0" maxOccurs="1">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element ref="ddms:security" minOccurs="1"
                                        maxOccurs="unbounded"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    <xs:element name="minimumAllowedInboundClassifications"
                        minOccurs="0" maxOccurs="1">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="ddms:security" minOccurs="1"
                                    maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="maximumAllowedInboundClassifications"
                        minOccurs="0" maxOccurs="1">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="ddms:security" minOccurs="1"
                                    maxOccurs="unbounded"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<xs:element name="endpoints" minOccurs="0"
    maxOccurs="1">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="endpoint" minOccurs="1"
                maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence minOccurs="0" maxOccurs="1">
                        <xs:element name="description" type="xs:string" />
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

## UNCLASSIFIED

```
</xs:sequence>
<xs:attribute name="id" type="xs:string"
  use="required"/>
<xs:attribute name="hostName" type="xs:string"
  use="optional"/>
<xs:attribute name="ipAddress" type="xs:string"
  use="optional"/>
<xs:attribute name="sendPort" type="xs:string"
  use="optional"/>
<xs:attribute name="receivePort" type="xs:string"
  use="optional"/>
<xs:attribute name="type" use="optional">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="client"/>
      <xs:enumeration value="server"/>
      <xs:enumeration value="both"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="protocol" use="optional">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="dsg2"/>
      <xs:enumeration value="dsg3"/>
      <xs:enumeration value="dsg4"/>
      <xs:enumeration value="rm"/>
      <xs:enumeration value="isse"/>
      <xs:enumeration value="http"/>
      <xs:enumeration value="https"/>
      <xs:enumeration value="ftp"/>
      <xs:enumeration value="sftp"/>
      <xs:enumeration value="tcp-socket"/>
      <xs:enumeration value="udp-socket"/>
      <xs:enumeration value="file"/>
      <xs:enumeration value="other"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
<xs:attribute name="pathInfo" type="xs:string"
  use="optional"/>
<xs:attribute name="certificate" type="xs:string"
  use="optional"/>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="id" type="xs:string" use="required"/>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
```

## UNCLASSIFIED

```
<xs:element name="files" minOccurs="0" maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:element name="schemas" minOccurs="0" maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:element name="schema" minOccurs="0"
maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="description" minOccurs="0"
maxOccurs="1"/>
</xs:sequence>
<xs:attribute name="id" type="xs:string" use="required"/>
<xs:attribute name="type" use="required">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="W3C"/>
<xs:enumeration value="RelaxNG"/>
</xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="file" type="xs:string"
use="required"/>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>

<xs:element name="schematrons" minOccurs="0" maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:element name="schematron" minOccurs="0"
maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="description" minOccurs="0"
maxOccurs="1"/>
</xs:sequence>
<xs:attribute name="id" type="xs:string" use="required"/>
<xs:attribute name="file" type="xs:string"
use="required"/>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>

<xs:element name="stylesheets" minOccurs="0" maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:element name="stylesheet" minOccurs="0"
maxOccurs="unbounded">
```

## UNCLASSIFIED

```
<xs:complexType>
  <xs:sequence>
    <xs:element name="description" minOccurs="0"
      maxOccurs="1"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required"/>
  <xs:attribute name="file" type="xs:string"
    use="required"/>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>

<xs:element name="others" minOccurs="0" maxOccurs="1">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="other" minOccurs="0"
        maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="description" minOccurs="0"
              maxOccurs="1"/>
          </xs:sequence>
          <xs:attribute name="id" type="xs:string" use="required"/>
          <xs:attribute name="file" type="xs:string"
            use="required"/>
          <xs:attribute name="type" type="xs:string"
            use="optional"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>

<xs:element name="domainPairs" minOccurs="1" maxOccurs="1">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="domainPair" minOccurs="1"
        maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="description" type="xs:string"
              minOccurs="0" maxOccurs="1" />
            <xs:element name="filterset" minOccurs="1"
              maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="description" minOccurs="0"
                    maxOccurs="1" />
                </xs:sequence>
                <xs:element name="guard" minOccurs="0"

```

## UNCLASSIFIED

```
maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:element name="options" type="dfc:options"
  minOccurs="0" maxOccurs="1" />
<xs:element name="xformConfig" minOccurs="0"
  maxOccurs="1">
<xs:complexType>
<xs:attribute name="xforms" type="xs:string"
  use="required"/>
<xs:attribute name="css" type="xs:string"
  use="optional"/>
<xs:attribute name="instance" type="xs:string"
  use="required"/>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="name" type="xs:string"
  use="required"/>
<xs:attribute name="version" type="xs:string"
  use="required"/>
<xs:attribute name="messageType" type="xs:string"
  use="optional"/>
<xs:attribute name="nativeSupport" type="xs:boolean"
  use="optional"/>
</xs:complexType>
<xs:key name="filtersetGuardOptionsKey">
<xs:selector xpath="dfc:options/dfc:option" />
<xs:field xpath="@name" />
</xs:key>
</xs:element>

<xs:element name="fromEndpoints" minOccurs="0"
  maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:element name="endpoint" minOccurs="1"
  maxOccurs="1">
<xs:complexType>
<xs:attribute name="ref" type="xs:string" />
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>

<xs:element name="toEndpoints" minOccurs="0"
  maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:element name="endpoint" minOccurs="1"
  maxOccurs="1">
<xs:complexType>
<xs:attribute name="ref" type="xs:string" />
</xs:complexType>
```

## UNCLASSIFIED

```
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>

<xs:element name="requires" minOccurs="0" maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:element name="require" minOccurs="1"
maxOccurs="unbounded">
<xs:complexType>
<xs:attribute name="filtersetRef" type="xs:string"
use="required"/>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="type" use="required">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="all"/>
<xs:enumeration value="any"/>
</xs:restriction>
</xs:simpleType>
</xs:attribute>
</xs:complexType>
</xs:element>

<xs:element name="dependencies" minOccurs="0"
maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:element name="dependency" minOccurs="1"
maxOccurs="unbounded">
<xs:complexType>
<xs:attribute name="filtersetRef" type="xs:string"
use="required"/>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>

<xs:element name="associates" minOccurs="0"
maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:element name="associate" minOccurs="1"
maxOccurs="unbounded">
<xs:complexType>
<xs:attribute name="filtersetRef" type="xs:string"
use="required"/>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
```

## UNCLASSIFIED

```
</xs:element>

<xs:element name="filters" minOccurs="1" maxOccurs="1">
  <xs:complexType>
    <xs:choice minOccurs="1" maxOccurs="unbounded">
      <xs:element name="xmlvalidator">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="description" type="xs:string"
              minOccurs="0" maxOccurs="1" />
            <xs:element name="options" type="dfc:options"
              minOccurs="0" maxOccurs="1" />
            <xs:element name="dependencies"
              type="dfc:dependencies" minOccurs="0"
              maxOccurs="1" />
            <xs:element name="filterDataInput"
              type="dfc:filterDataInput" minOccurs="1"
              maxOccurs="1"/>
          </xs:sequence>
          <xs:attribute name="id" type="xs:string"
            use="required"/>
          <xs:attribute name="schemaRef" type="xs:string"
            use="required" />
        </xs:complexType>

        <xs:key name="xmlvalidatorOptionsKey">
          <xs:selector xpath="dfc:options/dfc:option"/>
          <xs:field xpath="@name"/>
        </xs:key>
      </xs:element>
      <xs:element name="schematron">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="description" type="xs:string"
              minOccurs="0" maxOccurs="1" />
            <xs:element name="options" type="dfc:options"
              minOccurs="0" maxOccurs="1" />
            <xs:element name="dependencies"
              type="dfc:dependencies" minOccurs="0"
              maxOccurs="1" />
            <xs:element name="filterDataInput"
              type="dfc:filterDataInput"
              minOccurs="1" maxOccurs="1" />
          </xs:sequence>
          <xs:attribute name="id" type="xs:string"
            use="required"/>
          <xs:attribute name="schematronRef" type="xs:string"
            use="required" />
        </xs:complexType>
        <xs:key name="schematronOptionsKey">
          <xs:selector xpath="dfc:options/dfc:option"/>
          <xs:field xpath="@name"/>
        </xs:key>
      </xs:element>
      <xs:element name="xsltransformer">
```

## UNCLASSIFIED

```
<xs:complexType>
  <xs:sequence>
    <xs:element name="description" type="xs:string"
      minOccurs="0" maxOccurs="1" />
    <xs:element name="options" type="dfc:options"
      minOccurs="0" maxOccurs="1" />
    <xs:element name="dependencies"
      type="dfc:dependencies" minOccurs="0"
      maxOccurs="1" />
    <xs:element name="filterDataInput"
      type="dfc:filterDataInput"
      minOccurs="1" maxOccurs="1" />
  </xs:sequence>
  <xs:attribute name="id" type="xs:string"
    use="required" />
  <xs:attribute name="stylesheetRef" type="xs:string"
    use="required" />
</xs:complexType>
<xs:key name="xsltransformerOptionsKey">
  <xs:selector xpath="dfc:options/dfc:option"/>
  <xs:field xpath="@name"/>
</xs:key>
</xs:element>

<xs:element name="xslvalidator">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="description" type="xs:string"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="options" type="dfc:options"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="dependencies"
        type="dfc:dependencies" minOccurs="0"
        maxOccurs="1" />
      <xs:element name="filterDataInput"
        type="dfc:filterDataInput"
        minOccurs="1" maxOccurs="1" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string"
      use="required" />
    <xs:attribute name="stylesheetRef" type="xs:string"
      use="required" />
  </xs:complexType>
  <xs:key name="xslvalidatorOptionsKey">
    <xs:selector xpath="dfc:options/dfc:option"/>
    <xs:field xpath="@name"/>
  </xs:key>
</xs:element>

<xs:element name="wordchecker">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="description" type="xs:string"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="options" type="dfc:options"
```

## UNCLASSIFIED

```
    minOccurs="0" maxOccurs="1" />
<xs:element name="dependencies"
  type="dfc:dependencies" minOccurs="0"
  maxOccurs="1" />
<xs:element name="filterDataInput"
  type="dfc:filterDataInput"
  minOccurs="1" maxOccurs="1" />
</xs:sequence>
<xs:attribute name="id" type="xs:string"
  use="required" />
<xs:attribute name="file" type="xs:string"
  use="required" />
</xs:complexType>
<xs:key name="wordcheckerOptionsKey">
  <xs:selector xpath="dfc:options/dfc:option"/>
  <xs:field xpath="@name" />
</xs:key>
</xs:element>
<xs:element name="xmlnormalizer">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="description" type="xs:string"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="options" type="dfc:options"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="dependencies"
        type="dfc:dependencies" minOccurs="0"
        maxOccurs="1" />
      <xs:element name="filterDataInput"
        type="dfc:filterDataInput"
        minOccurs="1" maxOccurs="1" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string"
      use="required" />
  </xs:complexType>
  <xs:key name="xmlnormalizerOptionsKey">
    <xs:selector xpath="dfc:options/dfc:option"/>
    <xs:field xpath="@name"/>
  </xs:key>
</xs:element>
<xs:element name="classificationchecker">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="description" type="xs:string"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="options" type="dfc:options"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="dependencies"
        type="dfc:dependencies" minOccurs="0"
        maxOccurs="1" />
      <xs:element name="filterDataInput"
        type="dfc:filterDataInput"
        minOccurs="1" maxOccurs="1" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string"
```

## UNCLASSIFIED

```
        use="required" />
    </xs:complexType>
    <xs:key name="classificationcheckerOptionsKey">
        <xs:selector xpath="dfc:options/dfc:option" />
        <xs:field xpath="@name" />
    </xs:key>
</xs:element>
<xs:element name="viruschecker">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="description" type="xs:string"
                minOccurs="0" maxOccurs="1" />
            <xs:element name="options" type="dfc:options"
                minOccurs="0" maxOccurs="1" />
            <xs:element name="dependencies"
                type="dfc:dependencies" minOccurs="0"
                maxOccurs="1" />
            <xs:element name="filterDataInput"
                type="dfc:filterDataInput"
                minOccurs="1" maxOccurs="1" />
        </xs:sequence>
        <xs:attribute name="id" type="xs:string"
            use="required" />
    </xs:complexType>
    <xs:key name="viruscheckerOptionsKey">
        <xs:selector xpath="dfc:options/dfc:option" />
        <xs:field xpath="@name" />
    </xs:key>
</xs:element>
<xs:element name="digitalsignaturevalidator">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="description" type="xs:string"
                minOccurs="0" maxOccurs="1" />
            <xs:element name="urls" minOccurs="0"
                maxOccurs="1">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="url" minOccurs="1"
                            maxOccurs="unbounded">
                            <xs:complexType>
                                <xs:attribute name="value" type="xs:string"
                                    use="required"/>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="options" type="dfc:options"
    minOccurs="0" maxOccurs="1" />
<xs:element name="dependencies"
    type="dfc:dependencies" minOccurs="0"
    maxOccurs="1" />
<xs:element name="filterDataInput"
    type="dfc:filterDataInput"
```

## UNCLASSIFIED

```
        minOccurs="1" maxOccurs="1" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string"
      use="required" />
    <xs:attribute name="source" use="required">
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:enumeration value="internal"/>
          <xs:enumeration value="external"/>
          <xs:enumeration value="both"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>
  <xs:key name=" digitalsignaturevalidatorOptionsKey">
    <xs:selector xpath="dfc:options/dfc:option" />
    <xs:field xpath="@name" />
  </xs:key>
</xs:element>
<xs:element name="digitalsignureremover">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="description" type="xs:string"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="options" type="dfc:options"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="dependencies"
        type="dfc:dependencies" minOccurs="0"
        maxOccurs="1" />
      <xs:element name="filterDataInput"
        type="dfc:filterDataInput"
        minOccurs="1" maxOccurs="1" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string"
      use="required" />
  </xs:complexType>
  <xs:key name="digitalsignureremoverOptionsKey">
    <xs:selector xpath="dfc:options/dfc:option" />
    <xs:field xpath="@name" />
  </xs:key>
</xs:element>
<xs:element name="custom">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="description" type="xs:string"
        minOccurs="0" maxOccurs="1" />
      <xs:element name="files" minOccurs="0"
        maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="file" minOccurs="1"
              maxOccurs="unbounded">
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="description"
```

## UNCLASSIFIED

```
        type="xs:string" minOccurs="0"
        maxOccurs="1"/>
    </xs:sequence>
    <xs:attribute name="id" use="required" />
    <xs:attribute name="name" use="optional" />
    <xs:attribute name="type" use="optional"/>
    <xs:attribute name="file" use="required"/>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:key name="fileKey">
    <xs:selector xpath="dfc:file"/>
    <xs:field xpath="@id"/>
</xs:key>
</xs:element>
<xs:element name="options" type="dfc:options"
    minOccurs="0" maxOccurs="1" />
<xs:element name="dependencies"
    type="dfc:dependencies" minOccurs="0"
    maxOccurs="1" />
<xs:element name="filterDataInput"
    type="dfc:filterDataInput"
    minOccurs="1" maxOccurs="1" />

</xs:sequence>
<xs:attribute name="id" type="xs:string"
    use="required" />
<xs:attribute name="type" type="xs:string"
    use="required" />
</xs:complexType>
<xs:key name="customOptionsKey">
    <xs:selector xpath="dfc:options/dfc:option"/>
    <xs:field xpath="@name"/>
</xs:key>
</xs:element>
</xs:choice>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="id" type="xs:string"
    use="required" />
<xs:attribute name="fromDomainRef" type="xs:string"
    use="required" />
<xs:attribute name="toDomainRef" type="xs:string"
    use="required"/>
</xs:complexType>
<xs:key name="filtersKey">
    <xs:selector xpath="dfc:filters/*" />
    <xs:field xpath="@id" />
</xs:key>
<xs:keyref name="filtersDependencyKeyref"
    refer="dfc:filtersKey">
    <xs:selector
        xpath="dfc:filters/*/dfc:dependencies/dfc:dependency" />
```

## UNCLASSIFIED

```
<xs:field xpath="@filterRef" />
</xs:keyref>
<xs:keyref name="filtersDataInputDependencyKeyref"
refer="dfc:filtersKey">
<xs:selector
xpath="dfc:filters/*/dfc:filterDataInput" />
<xs:field xpath="@dependencyFilterRef" />
</xs:keyref>
</xs:element>
</xs:sequence>

<xs:attribute name="domain1Ref" type="xs:string"
use="required"/>
<xs:attribute name="domain2Ref" type="xs:string"
use="required"/>
</xs:complexType>

<xs:key name="filtersetKey">
<xs:selector xpath="dfc:filterset" />
<xs:field xpath="@id" />
</xs:key>

<xs:keyref name="requiresKeyref" refer="dfc:filtersetKey">
<xs:selector
xpath="dfc:filterset/dfc:requires/dfc:require"/>
<xs:field xpath="@filtersetRef" />
</xs:keyref>
<xs:keyref name="dependencyKeyref" refer="dfc:filtersetKey">
<xs:selector
xpath="dfc:filterset/dfc:dependencies/dfc:dependency"/>
<xs:field xpath="@filtersetRef" />
</xs:keyref>
<xs:keyref name="associateKeyref" refer="dfc:filtersetKey">
<xs:selector
xpath="dfc:filterset/dfc:associates/dfc:associate" />
<xs:field xpath="@filtersetRef" />
</xs:keyref>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="version" use="required">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="1.2.10" />
<xs:enumeration value="1.2.11" />
</xs:restriction>
</xs:simpleType>
</xs:attribute>
</xs:complexType>

<xs:key name="domainKey">
<xs:selector xpath="dfc:domains/dfc:domain"/>
<xs:field xpath="@id"/>
```

## UNCLASSIFIED

```
</xs:key>

<xs:keyref name="domain1Ref" refer="dfc:domainKey">
  <xs:selector xpath="dfc:domainPairs/dfc:domainPair"/>
  <xs:field xpath="@domain1Ref"/>
</xs:keyref>

<xs:keyref name="domain2Ref" refer="dfc:domainKey">
  <xs:selector xpath="dfc:domainPairs/dfc:domainPair"/>
  <xs:field xpath="@domain2Ref"/>
</xs:keyref>

<xs:keyref name="fromDomainRef" refer="dfc:domainKey">
  <xs:selector xpath="dfc:domainPairs/dfc:domainPair/dfc:filterset"/>
  <xs:field xpath="@fromDomainRef"/>
</xs:keyref>

<xs:keyref name="toDomainRef" refer="dfc:domainKey">
  <xs:selector xpath="dfc:domainPairs/dfc:domainPair/dfc:filterset"/>
  <xs:field xpath="@toDomainRef"/>
</xs:keyref>

<xs:key name="endpointKey">
  <xs:selector
    xpath="dfc:domains/dfc:domain/dfc:endpoints/dfc:endpoint" />
  <xs:field xpath="@id" />
</xs:key>
<xs:key name="schemaKey">
  <xs:selector xpath="dfc:files/dfc:schemas/dfc:schema" />
  <xs:field xpath="@id" />
</xs:key>
<xs:key name="schematronKey">
  <xs:selector xpath="dfc:files/dfc:schematrons/dfc:schematron" />
  <xs:field xpath="@id" />
</xs:key>
<xs:key name="stylesheetKey">
  <xs:selector xpath="dfc:files/dfc:stylesheets/dfc:stylesheet" />
  <xs:field xpath="@id" />
</xs:key>
<xs:key name="otherKey">
  <xs:selector xpath="dfc:files/dfc:others/dfc:other" />
  <xs:field xpath="@id" />
</xs:key>
<xs:keyref name="xmlvalidatorKeyref" refer="dfc:schemaKey">
  <xs:selector xpath="dfc:domainPairs/dfc:domainPair/
    dfc:filterset/dfc:filters/dfc:xmlvalidator" />
  <xs:field xpath="@schemaRef" />
</xs:keyref>
<xs:keyref name="schematronKeyref" refer="dfc:schematronKey">
  <xs:selector xpath="dfc:domainPairs/dfc:domainPair/
    dfc:filterset/dfc:filters/dfc:schematron" />
  <xs:field xpath="@schematronRef" />
</xs:keyref>
<xs:keyref name="xsltransformerKeyref" refer="dfc:stylesheetKey">
  <xs:selector xpath="dfc:domainPairs/dfc:domainPair/
```

## UNCLASSIFIED

```
    dfc:filterset/dfc:filters/dfc:xsltransformer" />
<xs:field xpath="@stylesheetRef" />
</xs:keyref>
<xs:keyref name="xslvalidatorKeyref" refer="dfc:stylesheetKey">
<xs:selector xpath="dfc:domainPairs/dfc:domainPair/
    dfc:filterset/dfc:filters/dfc:xslvalidator" />
<xs:field xpath="@stylesheetRef" />
</xs:keyref>
</xs:element>

<xs:complexType name="options">
<xs:sequence>
<xs:element name="option" minOccurs="1" maxOccurs="unbounded">
<xs:complexType>
<xs:simpleContent>
<xs:extension base="xs:string">
<xs:attribute name="name" use="required"/>
</xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>

<xs:complexType name="dependencies">
<xs:sequence>
<xs:element name="dependency" minOccurs="0"
    maxOccurs="unbounded">
<xs:complexType>
<xs:attribute name="filterRef" type="xs:string" use="required"/>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
<xs:complexType name="filterDataInput">
<xs:attribute name="location" use="required">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="dependency"/>
<xs:enumeration value="original"/>
<xs:enumeration value="none"/>
</xs:restriction>
</xs:simpleType>
</xs:attribute>
<xs:attribute name="dependencyFilterRef" type="xs:string"
    use="optional"/>
</xs:complexType>
</xs:schema>
```

## DigitalSignings.xsd

```
<?xml version="1.0"?>
```

## UNCLASSIFIED

```
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  targetNamespace="http://cie.jfcom.mil/schema/dfc"
  xmlns:dfc="http://cie.jfcom.mil/schema/dfc"
  elementFormDefault="qualified">
  <xs:import
    namespace="http://www.w3.org/2000/09/xmldsig#"
    schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-
      schema.xsd"/>
  <xs:element name="digitalSigning">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="fileEntry" minOccurs="1"
          maxOccurs="unbounded">
          <xs:complexType>
            <xs:attribute name="filename" type="xs:string"
              use="required"/>
            <xs:attribute name="hash" type="xs:string" use="required"/>
          </xs:complexType>
        </xs:element>
        <xs:element name="signed" type="xs:date" minOccurs="1"
          maxOccurs="1"/>
        <xs:element name="signedBy" type="xs:string" minOccurs="1"
          maxOccurs="1"/>
        <xs:element name="signerRole" minOccurs="1" maxOccurs="1">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="developer"/>
              <xs:enumeration value="submitter"/>
              <xs:enumeration value="evaluator"/>
              <xs:enumeration value="approver" />
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element ref="ds:Signature"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

## UNCLASSIFIED

### **DirtyCleanWords.xsd**

```
<?xml version="1.0"?>
<xss:schema xmlns:xss="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://cie.jfcom.mil/schema/dfc"
  elementFormDefault="qualified">
  <xss:element name="dirtyCleanWordLists">
    <xss:complexType>
      <xss:sequence>
        <xss:element name="description" type="xss:string" minOccurs="1"
          maxOccurs="1"/>
        <xss:element name="dirtyCleanWordList" minOccurs="0"
          maxOccurs="unbounded">
          <xss:complexType>
            <xss:sequence>
              <xss:element name="dirtyWords" minOccurs="1"
                maxOccurs="1">
                <xss:complexType>
                  <xss:attribute name="file" type="xss:string"
                    use="required"/>
                </xss:complexType>
              </xss:element>
              <xss:element name="cleanWords" minOccurs="0"
                maxOccurs="1">
                <xss:complexType>
                  <xss:attribute name="file" type="xss:string"
                    use="required"/>
                </xss:complexType>
              </xss:element>
              <xss:element name="ignoreCharacters" minOccurs="0"
                maxOccurs="1">
                <xss:complexType>
                  <xss:attribute name="file" type="xss:string"
                    use="required"/>
                </xss:complexType>
              </xss:element>
            </xss:sequence>
            <xss:attribute name="lang" type="xss:string" use="required"/>
          </xss:complexType>
        </xss:element>
      </xss:sequence>
    </xss:complexType>
  </xss:element>
</xss:schema>
```

**UNCLASSIFIED**

**ChangeLog.xsd**

```
<?xml version="1.0"?>
<xs:schema
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://cie.jfcom.mil/schema/dfc"
  xmlns:dfc="http://cie.jfcom.mil/schema/dfc"
  elementFormDefault="qualified">

  <xs:element name="changeLog">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="message" minOccurs="1" maxOccurs="unbounded">
          <xs:complexType>
            <xs:simpleContent>
              <xs:extension base="xs:string">
                <xs:attribute name="author" type="xs:string"
                  use="required" />
                <xs:attribute name="time" type="xs:dateTime"
                  use="required" />
              </xs:extension>
            </xs:simpleContent>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

## UNCLASSIFIED

### **Appendix B - References**

"Key Words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, Bradner, S., <http://www.ietf.org/rfc/rfc2119.txt?number=2119>

"XML 1.1 (Second Edition)", W3C Recommendation, 16 August 2006, Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, François Yergeau, John Cowan, ed., <http://www.w3.org/TR/2006/REC-xml11-20060816>

"XML Schema Part 0: Primer Second Edition", W3C Recommendation, 28 October 2004, David Fallside, Priscilla Walmsley, ed., <http://www.w3.org/TR/xmlschema-0>

"XML Schema Part 1: Structures Second Edition", W3C Recommendation, 28 October 2004, Henry S. Thompson, David Beech, Murray Maloney, Noah Mendelsohn, ed., <http://www.w3.org/TR/xmlschema-1>

"XML Schema Part 2: Datatypes Second Edition", W3C Recommendation, 28 October 2004, Paul V. Biron, Ashok Malhotra, ed., <http://www.w3.org/TR/xmlschema-2>

"Information technology -- Document Schema Definition Language (DSDL) -- Part 3: Rule-based validation – Schematron", ISO/IEC 19757-3:2006, [http://standards.iso.org/ittf/PubliclyAvailableStandards/c040833\\_ISO\\_IEC\\_19757-3\\_2006\(E\).zip](http://standards.iso.org/ittf/PubliclyAvailableStandards/c040833_ISO_IEC_19757-3_2006(E).zip)

"Information technology -- Document Schema Definition Language (DSDL) -- Part 2: Regular-grammar-based validation -- RELAX NG", ISO/IEC 19757-2:2003, [http://standards.iso.org/ittf/PubliclyAvailableStandards/c037605\\_ISO\\_IEC\\_19757-2\\_2003\(E\).zip](http://standards.iso.org/ittf/PubliclyAvailableStandards/c037605_ISO_IEC_19757-2_2003(E).zip)

"Information technology -- Document Schema Definition Language (DSDL) -- Part 2: Regular-grammar-based validation -- RELAX NG -- Amendment 1: Compact Syntax", ISO/IEC 19757-2:2003/Amd.1:2006, [http://standards.iso.org/ittf/PubliclyAvailableStandards/c040774\\_ISO\\_IEC\\_19757-2\\_2003\\_Amd\\_1\\_2006\(E\).zip](http://standards.iso.org/ittf/PubliclyAvailableStandards/c040774_ISO_IEC_19757-2_2003_Amd_1_2006(E).zip)

"Intelligence Community Information Security Marking Implementation Guide", Intelligence Community Metadata Working Group, Release 2.0, 30 April 2004

"XML-Signature Syntax and Processing", W3C Recommendation, 12 February 2002, Donald Eastlake, Joseph Reagle, David Solo, ed., <http://www.w3.org/TR/xmldsig-core>

"Canonical XML, Version 1.0", W3C Recommendation, 15 March 2001, John Boyer, <http://www.w3.org/TR/xml-c14n>